Воронежский государственный университет Факультет
прикладной математики, информатики и механики

Математика, информационные технологии, приложения

*Межвузовская научная конференция
молодых ученых и студентов*

Воронеж,
26 апреля 2023 г.

Воронеж
Издательско-полиграфический центр
«Научная книга»
2023

УДК 531(063)+51-7(063)
ББК 22.2я5+22.1я5
М34

Математика, информационные технологии, приложения : сборник трудов Межвузовской научной конференции молодых ученых и студентов, Воронеж, 27 апреля 2022 г. – Воронеж : Научная книга, 2022. – 292 с.

М34

ISBN 978-5-4446-****-*

Традиционно конференция является междисциплинарной. Важнейшая ее цель – ознакомление молодых специалистов с новыми веяниями в современной науке, а также обмен знаниями и достижениями в предложенных научных направлениях.

УДК 531(063)+51-7(063)
ББК 22.2я5+22.1я5

ISBN 978-5-4446-****-*

УДК 004.4

СОЗДАНИЕ ВЕБ-ЧАТА С АВТОПЕРЕВОДОМ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

Н. Д. Аверочкин

Воронежский государственный университет

Введение

Целью выполняемой работы является создание web-приложения, которое нацелено на облегчение межязыковой коммуникации, полностью автоматизирован перевод речи на выбранный язык во время видеоконференции, а также во время обмена текстовыми сообщениями.

1. Технические требования

Приложение включает следующие модули:

1. «Пользователь».
2. «Текстовый чат».
3. «Видеочат».

Модуль «Пользователь» обладает следующими возможностями:

1. Возможность выбора языка перевода.
2. Возможность включения/выключения микрофона.
3. Возможность включения/выключения субтитров с переведенной речью собеседника.
4. Возможность включения веб-камеры.
5. Возможность копирования ссылки, для создания конференции
6. Возможность принять/отклонить звонок другого пользователя.
7. Возможность выйти из приложения.

Модуль «Текстовый чат» обладает следующими свойствами:

1. Отправка сообщения в окно ввода.
2. Просмотр сообщений.
3. Оповещение о входе/выходе из чата другим пользователем

Модуль «Видеочат» обладает следующими свойствами:

1. Получение видеоизображения с веб-камеры другого пользователя.
2. Просмотр изображения со своей веб-камеры.

2. Анализ функциональности

На этапе проектирования была выделена одна роль для любого зарегистрировавшегося/авторизовавшегося человека: пользователь. У каждого пользователя изначально имеется доступ к общему текстовому чату и переводу сообщений в нём. А также, после входа в веб-приложение пользователь может разрешить к своей веб-камере и микрофону. Для того, чтобы увидеть изображение со своей веб-камеры и, в последствии, транслировать его и свой голос другому пользователю при помощи отправки своего номера конференции в общий чат для установления соединения при принятии вызова от другого пользователя (рис. 1).

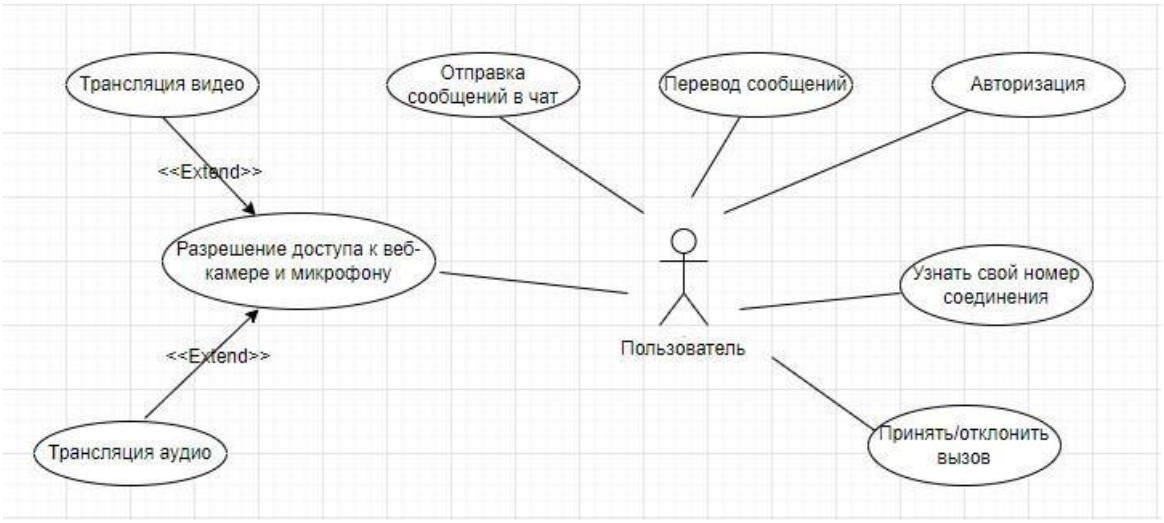


Рис. 1. Диаграмма использования базового функционала пользователя

Другой пользователь при получении сообщения с номером соединения в текстовый чат он может скопировать его и совершить звонок данному пользователю. Впоследствии получив оповещение о успешной/неуспешной отправке звонка. А также, получив изображение с веб-камеры пользователя адресата, при принятии вызова (рис. 2).

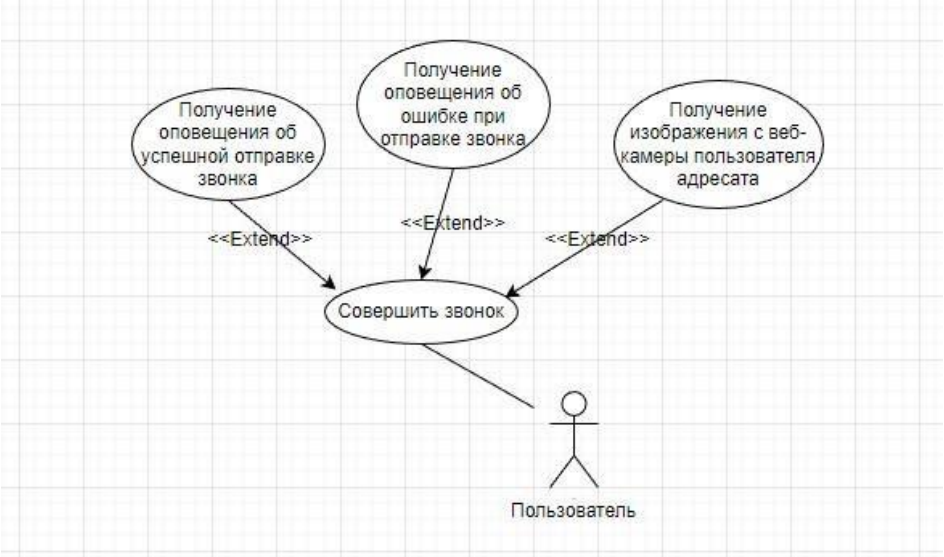


Рис. 2. Диаграмма установления видео-соединения.

После установления видео-соединения пользователи конференции пользователи могут использовать все возможности видео чата. Можно отключать/включать свою веб камеру и микрофон в приложении, сбросить вызов, а также использовать основную функцию – автоперевод речи собеседника субтитрами в реальном времени (рис. 3).

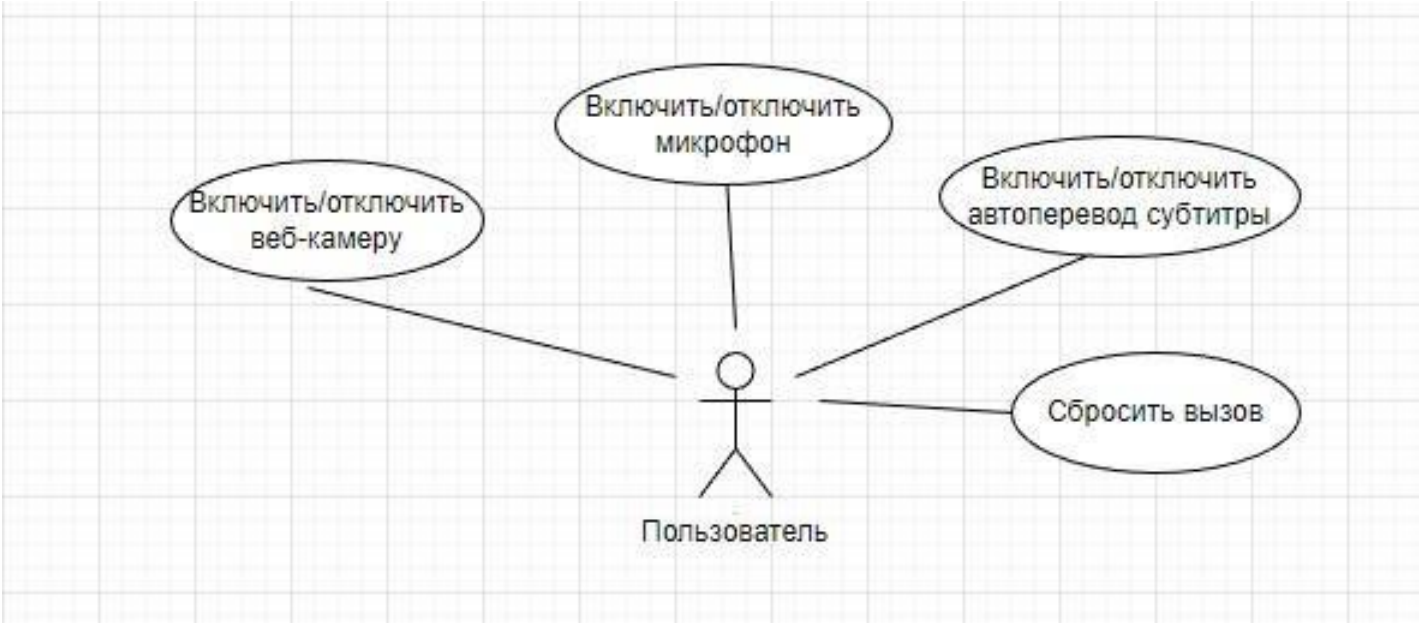


Рис. 3. Диаграмма основных действий при установке видео-соединения

Веб-приложение разрабатывалось в среде webstorm 2022. В качестве языка программирования были выбран JavaScript. Использовалась библиотека React для клиентской части, и серверное окружение nodejs для серверной части. React был выбран благодаря быстроты его работы, кроссплатформенности, а также большому количеству библиотек для него. Nodejs был выбран как оптимальный вариант написания небольшой и отказоустойчивой серверной части на языке javscript. Для хранения данных была выбрана документоориентированная система управления базами данных – MongoDb. Моделями, хранимыми в базе данных, являются сущности «TokenModel», «UserModel» (рис. 4).

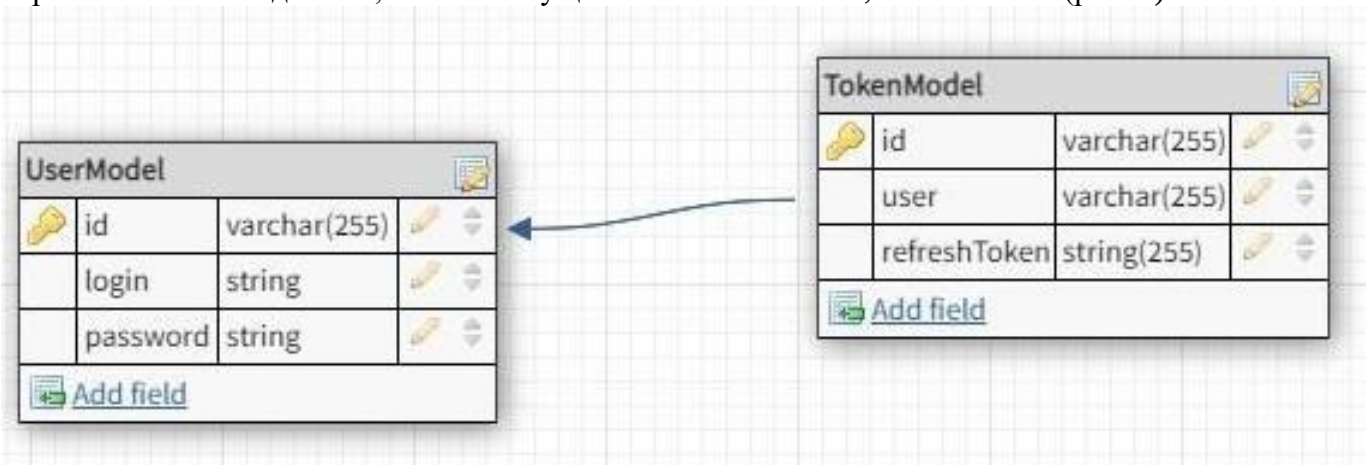


Рис. 4. Схема базы данных.

Заключение

В результате проделанной работы было создано веб-приложение, которое отвечает всем требованиям постановки задачи.

Полученное веб-приложение рассчитано на широкую аудиторию, и позволяет использовать его как в качестве обычного мессенджера, так и в качестве автоматизированной системы перевода для межязыковой коммуникации.

Литература

1. Макконнелл С. Совершенный код: учеб. пособие / С. Макконнелл 2-у изд., Москва : Русская редакция, 2010. 889 с.

2. Node.js в действии: учеб. пособие / К. Симпсон [и др.] ; под ред. Н. Райлих 2-е изд. СПб.: Питер, 2018. 432 с.

3. Тиленс Т.М. React в действии : учеб пособие / Т.М. Тиленс 1-е изд. СПб.: Питер, 2019. 368 с.

УДК *004.032.26*

НЕЙРОСЕТЕВОЕ РАСПОЗНАВАНИЕ РУКОПИСНОГО ТЕКСТА

К. И. Апасов, А. А. Журавлёв

Воронежский государственный университет

Введение

Распознавание рукописного текста является актуальной проблемой в области компьютерного зрения и машинного обучения. Эта проблема имеет множество практических применений, таких как автоматическое распознавание адресов на почтовых конвертах, распознавание рукописных подписей для авторизации и аутентификации, медицинских архивов, в том числе амбулаторных карт больных, а также распознавание других рукописных документов для архивирования и обработки. Традиционные методы распознавания рукописного текста обычно основываются на извлечении признаков из изображения символов и использовании классификаторов для их распознавания [1]. Однако, в последнее время нейросетевые подходы к распознаванию рукописного текста стали широко применяться благодаря своей способности к автоматическому извлечению признаков из изображений и глубокому обучению. Для решения проблемы распознавания рукописного текста, существует метод создания программы распознавания, которая использует нейросеть свёрточного типа. В данной работе для примера создадим программу распознавания рукописных цифр. Для создания такой программы мы будем использовать язык программирования C++. Выбор языка обуславливается его высокой производительностью, которая помогает в обработке большого количества обучающих материалов, кроссплатформенностью, что позволяет работать с проектом на различных операционных системах и аппаратных платформах, где важна масштабируемость, позволяющая добавлять новый функционал во время работы. Созданная программа должна представлять собой графический интерфейс с возможностью от руки создать изображение цифры, и нейросеть, реализованная в программе, должна определять её. Далее будет описан ход создания этого программного продукта.

1. Создание и обучение нейросети для распознавания рукописного текста

1.1. Создание интерфейса. Начальный этап работы.

Создадим графический интерфейс. Для создания использовался встроенный в C++ конструктор Windows Forms, в котором реализовали графическую панель и кнопку, при нажатии которой нарисованное на панели число сохранялось в bitmap буфер, далее производилось сжатие до рабочего разрешения 28x28.Потом мы начинали считывание данных. В bitmap каждый пиксель кодировался 4 битами (Red color, Green color, Blue Color, Alpha Channel), брался только первый бит, отбрасывая остальные 3, поскольку нам нужно было черно-белое изображение для удобства записи битов в массив (градации серого 0-255), а в черном цвете r, g и b биты имеют одинаковые значения, и записывали полученные биты в массив из 784 значений, этот массив назвали первым нейронным слоем. Далее программа создавала второй слой из 256 нейронов - скрытый слой, этот слой получает данные от входного слоя и на их основе извлекает признаки и использует их для классификации и передает данные

к 3-му слою - выходному. В выходном слое содержится 10 нейронов, каждый из которых отвечает за свою цифру от 0 до 9. Каждый нейрон этого слоя, так же, как и нейроны остальных слоев принимают определенные значения, будем называть его значением нейрона. Если значение конкретного нейрона на выходном слое будет близко к единице, будем считать, что нейросеть определила цифру, которая соответствует номеру нейрона. Для начала применим функции прямого распространения и активации. Каждый нейрон будет связан с каждым нейроном на следующем слое. За эти связи будут отвечать веса связи, которые масштабируют входные значения. Кроме Значений нейронов и связей, существует параметр смещений, управляющий тем, насколько активация будет работать даже в случае отсутствия сигналов на входах. В общем виде нейросеть будет представлять огромную функцию из всех этих переменных. Используя эту функцию и применяя на неё метод обратного распространения, мы можем добиться изменения ее значений, что и означает обучение нейросети. Для понимания дальнейшей работы нам требуется разобраться с математической частью нейронных сетей. В нашем случае, в работе нейросети используются четыре основных функции: функция активации, функция прямого распространения, функция ошибки и функция обратного распространения. Каждая из них имеет свои особенности и применяется для решения определенных задач.

1.2. Функция активации

Каждый нейрон в каждом нашем слое, должен иметь значения от -1 до 1, чтобы не выходить за границы этого диапазона, будем пользоваться функцией активации. В её роли будем использовать гиперболический тангенс (также известный как th(x)). Он является гладкой и нелинейной функцией [3].

Формула для гиперболического тангенса:

$$th(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (1)$$

Эта функция имеет S-образную форму, похожую на сигмовидную функцию, но симметричную относительно точки (0,0). Это означает, что входные значения отрицательны преобразуются в отрицательные выходные значения, а положительные входные значения - в положительные выходные значения. Одним из преимуществ гиперболического тангенса является его свойство градиентного сигнала. Он сохраняет градиентный сигнал, что означает, что градиент ошибки может эффективно распространяться через нейронную сеть в процессе обратного распространения. Другими словами, гиперболический тангенс позволяет нейронной сети обучаться более эффективно, поскольку он может помочь избежать проблемы затухающего или взрывающегося градиента, которые могут возникнуть при использовании других функций активации, таких как сигмоида. Обычно гиперболический тангенс используется в скрытых слоях нейронных сетей для улучшения обучения и получения более высокой точности предсказаний.

1.3. Функция прямого распространения

Функция прямого распространения является одной из ключевых функций в нейронных сетях и используется для передачи информации от входного слоя к выходному слою [4]. В

процессе прямого распространения каждый нейрон в слое получает входные данные от предыдущего слоя, выполняет взвешенное суммирование и применяет функцию активации для вычисления выходного значения. Формально, вычисление выходного значения нейрона i в слое j может быть описано следующей формулой [2]:

$$y_j^i = f\left(\sum_{k=1}^{N} \omega_k^i x_k^{j-1} + b_j^i\right) \tag{2}$$

где y_j^i- выходное значение нейрона i в слое j, x_k^{j-1} - входное значение нейрона k в предыдущем слое j-1, ω_k^i - весовой коэффициент между нейроном k в предыдущем слое j-1 и нейроном i в текущем слое j, b_j^i - смещение нейрона i в слое j, а f - функция активации.

1.4. Функция ошибки

Функция ошибки является мерой разницы между предсказанными и фактическими значениями на выходном слое. Она используется для оценки точности работы нейронной сети и определения того, как необходимо корректировать веса между слоями в процессе обучения. Одной из наиболее часто используемых функций ошибки является квадратичная функция ошибки, которая может быть записана следующим образом:

$$E = \frac{1}{2}\sum_{j=1}^{p}\left(y_j - \mathrm{d}_j\right)^2 \tag{3}$$

Где E - ошибка, y_j - выходное значение нейрона на выходном слое, d_j- фактическое значение на выходном слое.

1.5. Функция обратного распространения

Функция обратного распространения является методом обучения нейронной сети и используется для корректировки весов между слоями на основе ошибки, вычисленной на выходном слое. Обычно обучение нейронных сетей включает в себя несколько эпох, где каждая эпоха представляет собой полный цикл прохождения через все данные обучающей выборки. Каждый шаг обучения включает в себя передачу входных данных через сеть для получения выходных значений и вычисления функции потерь, затем обновление весов с помощью алгоритма обратного распространения ошибки. При обновлении весов с помощью обратного распространения ошибки используется градиентный спуск, который позволяет минимизировать функцию потерь путем изменения весов нейронов. Для этого используется градиент функции потерь, который показывает направление наискорейшего уменьшения потерь. Использование градиентного спуска с обратным распространением ошибки позволяет эффективно обучать нейронные сети на больших объемах данных и достигать высокой точности в задачах классификации, распознавания образов и других задачах машинного обучения.Конечная формула для обновления весов в слое j из слоя i может быть записана следующим образом [4]:

$$\omega_j^i = \omega_j^i - \alpha \frac{\partial E}{\partial \omega_j^i} \tag{4}$$

где α - скорость обучения (learning rate), а $\frac{\partial E}{\partial \omega_j^i}$ - частная производная функции ошибки E по весам ω_j^i. Чтобы вычислить эту частную производную, нужно применить правило цепочки. Предположим, что E зависит от выхода нейрона j:

$$\frac{\partial E}{\partial \omega_j^i} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net_j}{\partial \omega_j^i} \tag{5}$$

Здесь $\frac{\partial E}{\partial o_j}$ - частная производная функции ошибки E по выходу нейрона j, $\frac{\partial o_j}{\partial net_j}$ - частная производная функции активации по входу нейрона j (значение net_j перед функцией активации), а $\frac{\partial net_j}{\partial \omega_j^i}$ - частная производная входного сигнала нейрона j по весу ω_j^i.Частная производная $\frac{\partial E}{\partial o_j}$ зависит от выбранной функции ошибки E. Например, для среднеквадратичной ошибки $E = \frac{1}{2} \sum_{j=1}^{p} \left(y_j - \mathrm{d}_j \right)^2$ это выражение будет:

$$\frac{\partial E}{\partial o_j} = -\left(y_j - \mathrm{d}_j \right) \tag{6}$$

где y_j - фактическое значение выхода нейрона j, а d_j - предсказанное значение. Частная производная $\frac{\partial o_j}{\partial net_j}$ также зависит от выбранной функции активации. Для гиперболического тангенса, используемого в данной нейросети, это будет:

$$\frac{\partial o_j}{\partial net_j} = 1 - o_j^2 \tag{7}$$

Наконец, частная производная $\frac{\partial net_j}{\partial \omega_j^i}$ равна значению входного сигнала нейрона i:

$$\frac{\partial net_j}{\partial \omega_j^i} = o_i \tag{8}$$

Подставляя все значения, получаем:

$$\frac{\partial E}{\partial \omega_j^i} = -\left(y_j - \mathrm{d}_j \right) \cdot \left(1 - o_j^2 \right) \cdot o_i \tag{9}$$

Используя эту формулу, мы можем обновить веса между слоями и повторить процесс обучения

до достижения необходимой точности.

1.6. Финальный этап работы. Обучение нейросети

Далее на языке C++, используя базовую методологию объектно-ориентированного программирования, был полностью с нуля реализован класс "Нейрон", так же были реализованы все вышеперечисленные математические функции. В следствие связывания между собой всех классов была получена готовая, но еще не обученная нейронная сеть. Далее требовалось произвести ее обучение и тестирование, используя MNIST (Modified National Institute of Standards and Technology) — это набор данных рукописных цифр, который часто используется в качестве тестового набора для обучения нейронных сетей для распознавания изображений. Он состоит из 60 000 обучающих изображений и 10 000 тестовых изображений, каждое изображение имеет размер 28x28 пикселей. Для обучения нейронной сети на базе MNIST мы используем метод обратного распространения и функцию ошибки, описанные выше. Выполнив обучение осталось только связать нашу программу с ранее подготовленным графическим интерфейсом и выполнить тестирование для оценки точности.

2. Результаты и обсуждение

Для оценки результатов работы нейросети мы использовали метрику точности, которая рассчитывается как отношение количества правильно распознанных изображений к общему количеству изображений в тестовом наборе данных [5]. Для проверки нашей нейросети мы так же использовали набор данных MNIST. После обучения нашей нейросети на тренировочном наборе данных мы проверили ее точность на валидационном наборе данных и обнаружили, что точность составляет примерно 96,5%. Затем мы провели окончательную проверку нашей нейросети на тестовом наборе и получили точность около 95,2%. Следует отметить, что наша нейросеть достаточно быстро обрабатывает изображения, что позволяет ее использовать в режиме реального времени на устройствах с ограниченными ресурсами. Однако, при использовании нейросети на слабых устройствах или при обработке больших объемов данных, возможно снижение скорости обработки. Также мы обнаружили, что точность распознавания зависит от качества изображения. Изображения с низким качеством, сильно зашумленные или содержащие слишком тонкие линии, могут быть распознаны неправильно. В будущем мы планируем улучшить точность нашей нейросети путем использования более сложных алгоритмов обработки изображений и улучшения качества данных.

.

Заключение

В этой работе было описано создание нейросети для распознавания рукописного текста, используя набор данных MNIST и метод прямого распространения. Мы также разработали графический интерфейс, позволяющий пользователю рисовать изображения и отправлять их на распознавание. Мы провели тестирование нашей нейросети на наборе данных и получили точность около 95,2%, что является хорошим результатом. Однако, мы также обнаружили, что точность распознавания зависит от качества изображения, и мы планируем улучшить точность нашей нейросети в будущем. Последующие исследования могут включать улучшение точности распознавания при работе с изображениями низкого качества, оптимизацию алгоритмов обработки изображений и исследование возможностей использования нейросетей для распознавания других типов рукописных символов. Научный руководитель – преподаватель,

кафедры ERP-систем и бизнес-процессов ВГУ, Дергачёв Юрий Аркадьевич.

Литература

1. Бройдо, В. Л. Вычислительные системы, сети и телекоммуникации / В. Л. Бройдо. — СПб.: Питер, 2004. – 703 с.
2. LeCun, Y. The MNIST database of handwritten digits. – Режим доступа: http://yann.lecun.com/exdb/mnist.
3. Гаршин, А.А., Солдатова, О.П. Автоматизированная система распознавания рукописных цифр на основе свёрточной нейронной сети // Свидетельство об официальной регистрации программ для ЭВМ №2010610988 по заявке №2009616812 от 1 декабря 2009 года. Зарегистрировано в Реестре программ для ЭВМ 1 февраля 2010 года.
4. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. – М.: Вильямс, 2006. – 1104 с.
5. Волкова, М. А. Методы обработки и распознавания изображений / М. А. Волкова, В. Р. Луцив. – СПб: Университет ИТМО, 2016. – 40 с.

УДК 004.8

МЕТОД ГЕНЕРАЦИИ ИДЕЙ ДЛЯ ПРИЛОЖЕНИЯ ЭКСПЕРТИЗЫ КОЛЛЕКТИВНЫХ ИДЕЙ

А. В. Астахова

Воронежский государственный университет

Введение

Для того, чтобы любая организация оставалась компанией инновационного уровня, ей необходимо постоянно пересматривать и развертывать новые идеи и предложения. Актуальность работы заключается в необходимости создания и формирования новых идей для сокращения или устранения пробелов в процессе работы компаний.

В работе рассматривается возможность расширения приложения экспертизы коллективных идей. Данное расширение позволит генерировать новые идеи на основе предложений сотрудников внутри компании.

1. Анализ существующего приложения экспертизы коллективных идей

Любые компании находятся в поиске бизнес-идей, которые позволят им получать больше прибыли, уменьшить издержки производства, а также выигрывать конкуренцию. Большое количество рациональных идей по улучшению процессов работы могут предло-жить сотрудники компании.

В результате, спроектировано и реализовано приложение экспертизы коллективных идей, которое организовывает и развертывает предложения сотрудников компании в режиме реального времени. Структурно-программный комплекс данного приложения включает в себя компоненты, представленные на рис. 1.

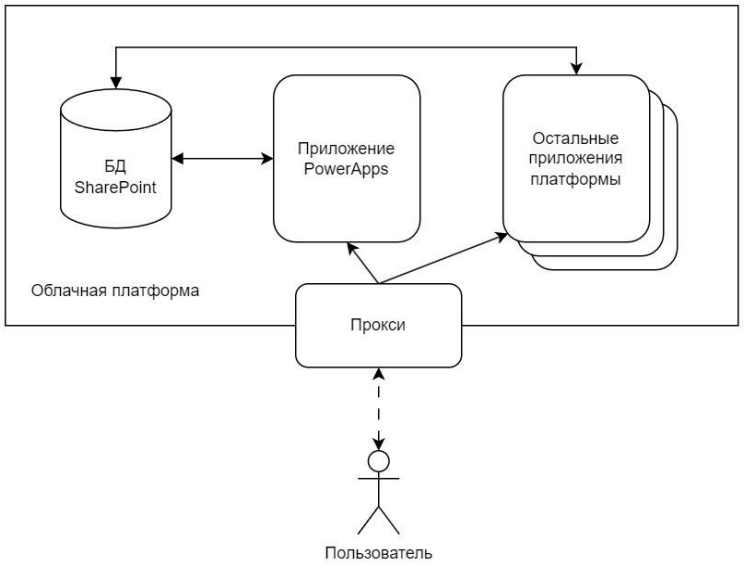


Рис. 1. Компоненты приложения экспертизы коллективных идей

В приложении, реализующем организацию коллективных идей, пользователь имеет следующие возможности:

- авторизоваться в приложении;
- добавить новые идеи;
- редактировать и удалить свои идеи;
- контролировать статус реализации своих идей;
- контролировать статус реализации идей других пользователей;
- просмотреть и осуществить поиск идей для реализации;
- выбрать идеи для реализации;
- комментировать существующие идеи;
- получать комментарии к своим идеям.

Реализованное приложение позволяет осуществлять сбор идей, которые в дальнейшем могут быть обработаны и использованы для генерации новых идей.

2. Метод генерации новых идей

2.1. Анализ алгоритма генерации идей

Помимо организованного сбора идей в приложении существует необходимость в развитии решения основной и главной задачи данного процесса — создание идей.

Часть алгоритма генерации идей можно выделить из метода Фрица Цвикки «Морфологический анализ» или «Матрица возможностей». Метод включает создание матрицы со строками и столбцами, где каждая строка представляет отдельный аспект проблемы, а каждый столбец представляет возможное решение. Например, если проблема заключается в том, как улучшить обслуживание клиентов компании, строки могут быть такими аспектами, как скорость ответа, удобство и качество решения. В столбцах могут быть представлены возможные решения, такие как увеличение числа сотрудников, повышение квалификации или улучшение каналов связи.

Процесс морфологического анализа включает в себя заполнение матрицы различными комбинациями решений каждого аспекта проблемы, генерируя большое количество возможных решений. После того, как матрица заполнена, решения оцениваются на основе их осуществимости, рентабельности и влияния на проблему.

Для реализации приложения, предоставляющего возможность генерировать идеи из уже существующих, адаптируем данный алгоритм следующим образом. Для упрощения модели, предположим, что идеи имеют определенную структуру. Пример представлен на рис. 2.

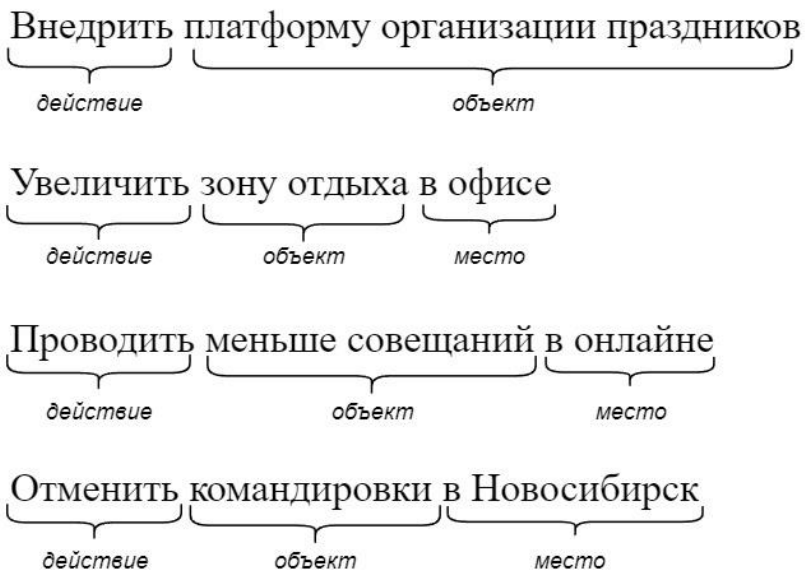


Рис. 2. Пример выделения структуры в тексте

В предложенном примере из каждой идеи выделено действие, объект и место. Можно использовать и более сложные структуры. На основе выделенных структурных единиц возможно создать новую идею, взяв действие «Внедрить» из первой идеи, объект «зону отдыха» из второй идеи и место «в онлайне» из третьей идеи. Получим новую идею «Внедрить зону отдыха в онлайне». Очевидно, что не каждая идея, полученная комбинированием действий и объектов, будет осмысленной. Отбор осмысленных идей отдельная задача, которую может выполнять пользователь или программный модуль с определенным алгоритмом.

Существует несколько алгоритмов и методов извлечения структуры из предложений, включая структуру объект-действие-место.

1. Анализ зависимостей.
2. Распознавание именованных объектов.
3. Библиотеки обработки естественного языка (NLP).
4. Методы машинного обучения.

Широко распространена реализация библиотеки обработки естественного языка Natasha на Python, которая позволяет проводить синтаксический и морфологический анализ, выделяя основные сущности.

2.2. Выделение ключевых слов

В условиях возможного потока вариаций идей, появляется необходимость фильтровать их автоматически с помощью алгоритмов или методов. Одним из возможных подходов к выделению смысла текста и оценке его релевантности является способ извлечения ключевых слов. Ключевыми словами в общем понимании называют важные слова, которые могут выступать в качестве метаинформации для задачи поиска и классификации идей.

Различают понятия ключевые слова и ключевые словосочетания (фразы). Ключевые фразы представляют собой сочетание двух или более слов, следующих друг за другом в тексте, либо разделенные другими языковыми единицами. При помощи отдельных ключевых слов затруднительно выразить основной смысл содержимого, в результате, преимуществом для формирования идей обладает именно понятие ключевой фразы.

Для извлечения ключевых слов из текста существуют различные алгоритмы. Широко распространены Rake и YAKE, способные обрабатывать тексты на русском языке.

Rake – это алгоритм извлечения ключевых слов, который использует список стоп-слов для идентификации и извлечения ключевых слов из заданного текста. Он работает, разбивая текст на отдельные слова, а затем идентифицируя ключевые слова-кандидаты на основе их частоты и совпадения с другими словами в тексте. Затем Rake оценивает каждое ключевое слово-кандидат на основе его частоты и совпадения с другими ключевыми словами, используя метрику, называемую «степень центральности». Эта метрика измеряет, насколько важно данное ключевое слово для всего текста, опираясь на то, в каком количестве других ключевых слов оно встречается. Конечным результатом является список ключевых слов-кандидатов с наивысшей оценкой.

YAKE – это более современный алгоритм извлечения ключевых слов, в котором используется подход, отличный от RAKE. Вместо того, чтобы полагаться на стоп-слова для определения ключевых слов-кандидатов, YAKE использует статистическую модель, основанную на распределении n-грамм (групп из n последовательных слов) в тексте. YAKE также использует модель машинного обучения для фильтрации нерелевантных n-грамм и ранжирования оставшихся ключевых слов-кандидатов. Конечным результатом является список ключевых слов-кандидатов с наивысшей оценкой вместе с их оценками релевантности.

Применив один из данных алгоритмов в модуле выделения ключевых слов появится возможность получить соответствие идеи и набора её ключевых слов. Это позволит пользователям выбирать ключевое слово как категорию, по которой будут генерироваться идеи.

2.3. Применение алгоритма генерации идей

После выделения структурных единиц в идеях, составим три множества: множество действий, объектов и мест из всех идей. После этого, будем выбирать случайные элементы из этих множеств, и соединять их в одно предложение. Таким образом, будут формироваться новые идеи.

Возможна модификация алгоритма, в котором будет добавлен шаг кластеризации перед генерацией новых идей, для того чтобы формировать новые идеи на основе структур из разных кластеров. Такой подход может помочь достичь генерации более разнообразных идей. Однако для этого необходимо подобрать метрику, которая позволяла бы кластеризовать идеи по контексту.

3. Описание модулей приложения генерации идей

Приложение должно иметь возможность получать идеи в виде строковых значений из базы данных уже реализованного приложения. Эти идеи должны сохраняться в новой базе данных, которая будет иметь возможность ассоциировать идеи с их разделением на структурные единицы, а также хранить ключевые слова для каждой идеи.

Идеи требуют предобработки, с фильтрацией не подходящих по структуре идей. За это будет отвечать модуль предварительной обработки идей. Затем в модуле выделения структур идеи будут разделены на структурные единицы, а также будут выделены ключевые слова. В модуле генерации структурные единицы будут комбинироваться случайным образом, и в качестве результата будут сформированы новые идеи. Схема взаимодействия модулей приложения представлена на рис. 3.

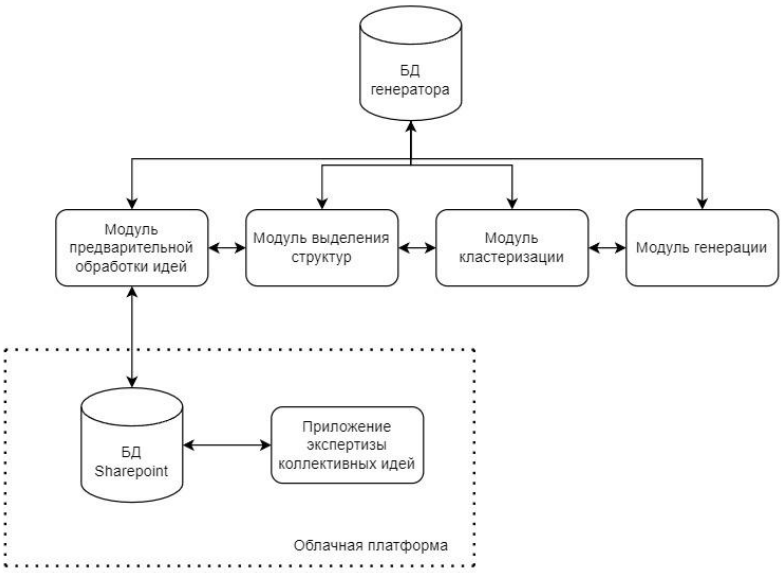


Рис. 3. Схема взаимодействия модулей приложения

Заключение

Применение метода генерации идей для приложения экспертизы коллективных идей позволит реализовать большое количество возможных вариантов предложений по рабочим процессам.

В результате работы был рассмотрен алгоритм генерации новых идей на основе существующих. Описан способ разделения идей на структурные единицы с последующим комбинированием их в новое предложение. Рассмотрена возможность модификации алгоритма добавлением кластеризации идей для генерации уникальных идей.

Литература

1. Астраханцев Н. А. Автоматическое извлечение терминов из коллекции текстов предметной области с помощью Википедии. / Н. А. Астраханцев // Труды Института системного программирования РАН. – 2014. – № 4. – С. 7–20.

2. Воронина, И. Е. Алгоритмы определения семантической близости ключевых слов по их окружению в тексте / И. Е. Воронина, А. А. Кретов, И. В. Попова // Вестник Воронежского государственного университета. – 2010. – № 1. – С. 148–153.

3. Грин, Э. Креативность в Паблик Рилейшнз / Э. Грин; пер. с англ. под ред. А. Н. Андреевой. – СПб.: Издательский Дом «Нева», 2003. – 224 с.

4. Гринева М. Анализ текстовых документов для извлечения тематически сгруппированных ключевых терминов / М. Гринева, М. Гринев // Труды Института системного программирования РАН. – 2009. – С. 155–165.

5. Дербенцев Д. Д. Практика применения морфологического подхода Фрица Цвикки в России и за рубежом. Замысел его применения при проектировании систем стратегического планирования и управления в России / Д. Д. Дербенцев, З. А Кучкаров // Международная научная конференция Нижегородск. гос. ун-та и научно-исследовательского центра физико-технической информатики. – 2019. – № 1. – С. 165–176.

6. Стожок Е. В. Ключевые слова как элементы терминосистем / Е. В. Стожок // Вестник Бурятск. гос. ун-та. – 2009. – № 11. – С. 101-104.

УДК 004.89

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ, СПОСОБНОГО РАСПОЗНАВАТЬ ПТИЦ С ИСПОЛЬЗОВАНИЕМ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

С. Б. В. Балок

Воронежский государственный университет

Введение

Во время поездок в лес часто случается, что мы хотим знать название птицы или, по крайней мере, класс, к которому она принадлежит, но из-за отсутствия глубоких знаний в этой области мы часто остаемся без ответа на этот вопрос. Орнитологи в своих исследованиях были бы рады иметь инструмент, способный быстро и эффективно классифицировать птиц, это, безусловно, было бы огромной экономией не только времени, но еще и человеческих, и финансовых ресурсов. Студентам, изучающим биологию животных, также может понадобиться инструмент для быстрой классификации разных видов птиц по фото, с превосходной точностью.

Эти различные, выше представленные причины подтверждают необходимость существования приложения, способного эффективно и быстро классифицировать различные виды птиц по фотографии.

Основной целью данной статьи является описание приложения, которое позволит аутентифицированным пользователям быстро и эффективно определять класс, к которому принадлежит птица на фотографии.

В статье будут представлены различные технологии, которые позволят реализовать веб-приложение, уделив особое внимание фреймворку Flask для разработки самого веб-приложения и библиотекам Tensorflow и Keras для построения модели на основе сверточных нейронных сетей, которые будут представлять собой интеллектуальную часть приложения.

1. Архитектура приложения

1.1. Модули приложения

Для реализации данного приложения был использован язык Python в среде Google Colab и Pycharm. Данная работа состоит из двух основных частей: интеллектуальной части, которая касается реализации алгоритма искусственного интеллекта на основе сверточных нейронных сетей, и визуальной части, которая представляет собой веб-приложение с пользовательским интерфейсом, управляющим аутентификацией по логину и паролю, предоставляемой базой данных MongoDB. Различные используемые модули представлены на рисунке 1:

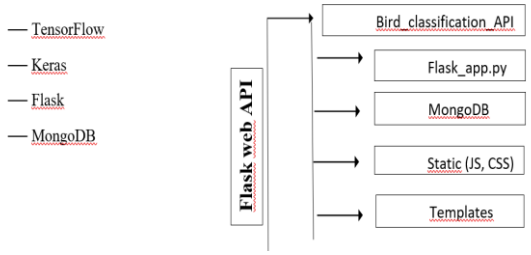


Рис. 1. Модули приложения

1.2. Технологии реализации интеллигентной части

Для построена алгоритма машинного обучения, была использована модель VGG16, которая представляет собой модель, основанную на искусственных нейронных сетях, способных распознавать объекты разных классов с очень хорошей точностью.
VGG16 — это модель модуля Keras фреймворка Tensorflow.

```
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
```

Рис. 2. Основные Библиотеки

Сначала необходимо импортировать нужные библиотеки как показано на рисунке 2.

Дальше важно дать размера картинок, был выбран 224×224, и разделяем датасет на тренировочные, тестовые и валидационные выборки.

```
IMAGE_SIZE = [224, 224]
train_directory = '/content/drive/MyDrive/Новая папка/train'
test_directory = '/content/drive/MyDrive/Новая папка/test'
val_directory = '/content/drive/MyDrive/Новая папка/valid'
```

Рис. 3. Разделение набора данных

```
# добавить слой предварительной обработки в начало VGG
vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)

# не надо тренировать существующие веса
for layer in vgg.layers:
    layer.trainable = False
```

Рис. 4. Добавление предварительно обученного слоя в начале VGG

Поскольку модель предварительно обучена, не нужно полностью переобучать ее, просто нужно обучить ее на новый набор данных, поэтому добавляем предварительно обученный слой в начало нейронной сети VGG.

```
# наши слои
x = Flatten()(vgg.output)
# x = Dense(1000, activation='relu')(x)
prediction = Dense(len(folders), activation='softmax')(x)

# создание модели
model = Model(inputs=vgg.input, outputs=prediction)
```

Рис. 5. Построение слоев и самой модели

Теперь построим слои и создадим модель (рисунок 5).

block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Рис. 6. Архитектура модели

Нейронная сеть с пяти слоями была создана (рисунок 6).

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set = train_datagen.flow_from_directory(train_directory,
                                                 target_size = (224, 224),
                                                 batch_size = 150,
                                                 class_mode = 'categorical')

test_set = test_datagen.flow_from_directory(test_directory,
                                            target_size = (224, 224),
                                            batch_size = 150,
                                            class_mode = 'categorical')
```

```
Found 32627 images belonging to 400 classes.
Found 1732 images belonging to 400 classes.
```

Рис. 7. Подготовка изображения

В качестве функции потерь, была использована categorical_crossentropy, как optimizer был выбран метод adam, для оценки модели выбор пал на метрику Accuracy.

Теперь подготовим данные для предварительной обработки, чтобы извлечь изображения каждого класса для обучающей и тестовой выборки (рисунок 7)

Перейдем к обучению модели на 10 epoch (рисунок 8).

```
r = model.fit_generator(
    training_set,
    validation_data=test_set,
    epochs=10,
    steps_per_epoch=len(training_set),
    validation_steps=len(test_set)
)
```

Рис. 8. Обучение модели

```
<ipython-input-11-5932e3024fde>:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future vers
  r = model.fit_generator(
Epoch 1/10
218/218 [==============================] - 8719s 40s/step - loss: 3.3629 - accuracy: 0.3792 - val_loss: 7.3823 - val_accuracy: 0.4873
Epoch 2/10
218/218 [==============================] - 485s 2s/step - loss: 1.2489 - accuracy: 0.7111 - val_loss: 6.7967 - val_accuracy: 0.5468
Epoch 3/10
218/218 [==============================] - 492s 2s/step - loss: 0.8290 - accuracy: 0.7976 - val_loss: 6.6396 - val_accuracy: 0.5687
Epoch 4/10
218/218 [==============================] - 489s 2s/step - loss: 0.5937 - accuracy: 0.8498 - val_loss: 6.4009 - val_accuracy: 0.5912
Epoch 5/10
218/218 [==============================] - 488s 2s/step - loss: 0.4333 - accuracy: 0.8886 - val_loss: 6.2719 - val_accuracy: 0.6039
Epoch 6/10
218/218 [==============================] - 487s 2s/step - loss: 0.3716 - accuracy: 0.9039 - val_loss: 6.3137 - val_accuracy: 0.6039
Epoch 7/10
218/218 [==============================] - 486s 2s/step - loss: 0.3088 - accuracy: 0.9183 - val_loss: 6.5031 - val_accuracy: 0.5791
Epoch 8/10
218/218 [==============================] - 492s 2s/step - loss: 0.2557 - accuracy: 0.9282 - val_loss: 6.4072 - val_accuracy: 0.5912
Epoch 9/10
218/218 [==============================] - 498s 2s/step - loss: 0.2257 - accuracy: 0.9369 - val_loss: 6.3541 - val_accuracy: 0.6097
Epoch 10/10
218/218 [==============================] - 492s 2s/step - loss: 0.2010 - accuracy: 0.9439 - val_loss: 6.3598 - val_accuracy: 0.6005
```

Рис. 9. Процесс обучение модели и оценка точности

После обучения модели, видно, что оценка точности в порядке 94% на тестовые выборки и в порядке 60% на валидационные выборки в то же время, видно, что функция потерь на тестовые выборки в порядке 0,2 а на валидационные выборки в порядке 6,3 (рисунок 9)

```
# loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
```

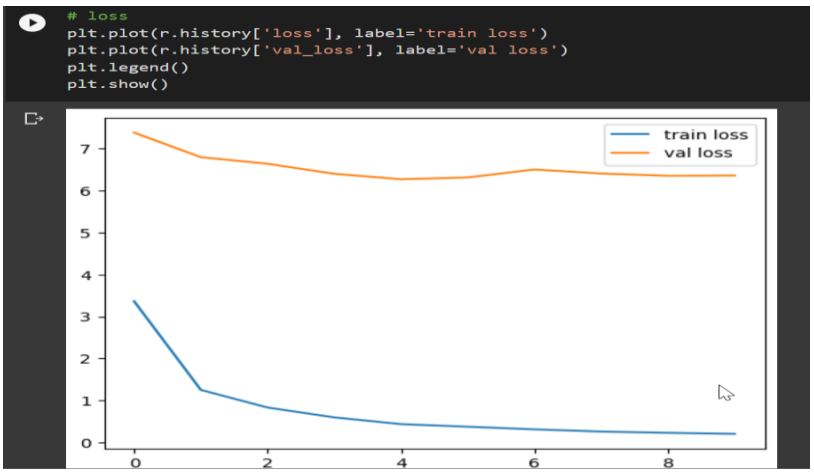


Рис. 10. Функция потерь

Можно графически наблюдать эволюцию (рисунок 10)

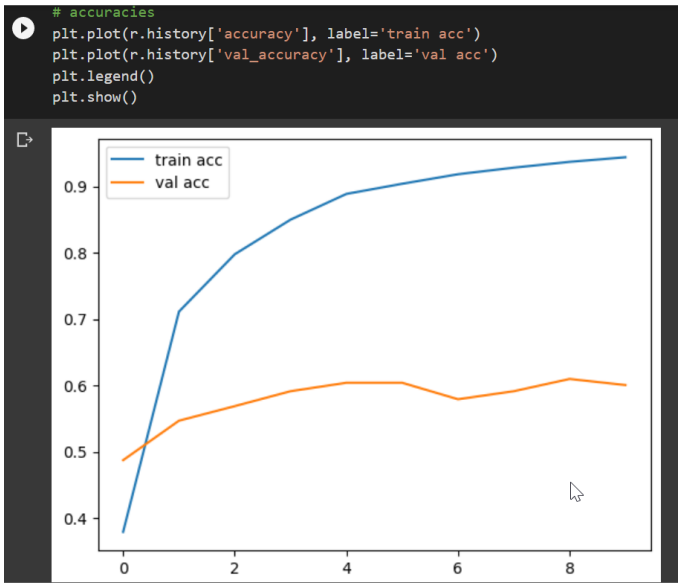


Рис. 11. Точность

Можно повысить точность модели на валидационные выборки, например, увеличив количество эпох.

Когда удовлетворяет точностью моделей, необходимо ее сохранить.

(рисунок 11)

1.3. Структурная схема системной архитектуры

Рис. 12. Структурная схема системной архитектуры

Весь проект можно резюмировать схемой, представленной на рис. 12

2. Реализация веб-приложения

Цель этого приложения — предоставить доступ аутентифицированным пользователям, которые находятся в базе данных. После аутентификации пользователь напрямую попадает на главную страницу (home.html), где ему будет предложено загрузить фотографию птицы, затем нажать кнопку загрузить (Upload), которая перенаправит на страницу результата (success.html). Здесь будет видно результат предсказания модель машинного обучения с хорошей точностью, которой измеряется метрикой Accuracy

Для реализации веб-приложения на Flask, нужно сначала импортировать нужные библиотеки (рисунок 13)

```
from flask_pymongo import PyMongo
import bcrypt



import os
import uuid
import urllib
from keras.models import load_model

from flask import Flask, render_template, request, \
    jsonify,  url_for, session, redirect
from keras.preprocessing.image import load_img, img_to_array
```

Рис. 13. Нужные библиотеки

```
app = Flask(name)

app.config["SECRET_KEY"] = "super-secret"


app.config['MONGO_DBNAME'] = 'mongologinexample'
app.config['MONGO_URI'] = 'mongodb://localhost:27017/mongologinexample'

mongo = PyMongo(app)
model = load_model('BC.h5')
```

Рис. 14. Подключения Flask и MongoDB

Дальше нужно подключить Flask и база данных MongoDB и сразу же импортировать нашу модель, которую мы ранее сохранили под названием BC.h5 (рисунок 14).

Данное веб-приложение содержит 4 основные страницы: home, login, register и success.

Страница register позволяет пользователю создать аккаунт и зарегистрироваться в нашей базе данных (рисунок 15).

```
@app.route('/register', methods=['POST', 'GET'])
def register():
    if request.method == 'POST':
        users = mongo.db.users
        existing_user = users.find_one({'name': request.form['username']})
        if existing_user is None:
            hashpass = bcrypt.hashpw(request.form['pass'].encode('utf-8'), bcrypt.gensalt())
            users.insert_one({'name': request.form['username'], 'password': hashpass})
            session['username'] = request.form['username']
            return redirect(url_for('index'))
        return 'That username already exists!'
    return render_template('register.html')
```

Рис. 15. Создание пут для страницы register

```
@app.route('/login', methods=['POST'])
def login():
    users = mongo.db.users
    login_user = users.find_one({'name': request.form['username']})

    if login_user:
        if bcrypt.hashpw(request.form['pass'].encode('utf-8'), login_user['password']) == login_user['password']:
            session['username'] = request.form['username']
        return redirect(url_for('index'))
    return 'Invalid username/password combination'
```

Рис. 16. Создание пут для страницы login

Страница login позволяет пользователю, который уже создал аккаунт, аутентифицироваться и входить на главную страницу (рисунок 16)

После регистрации пользователь автоматически добавляется в базу данных, в разделе users, как показано на рис. 17.

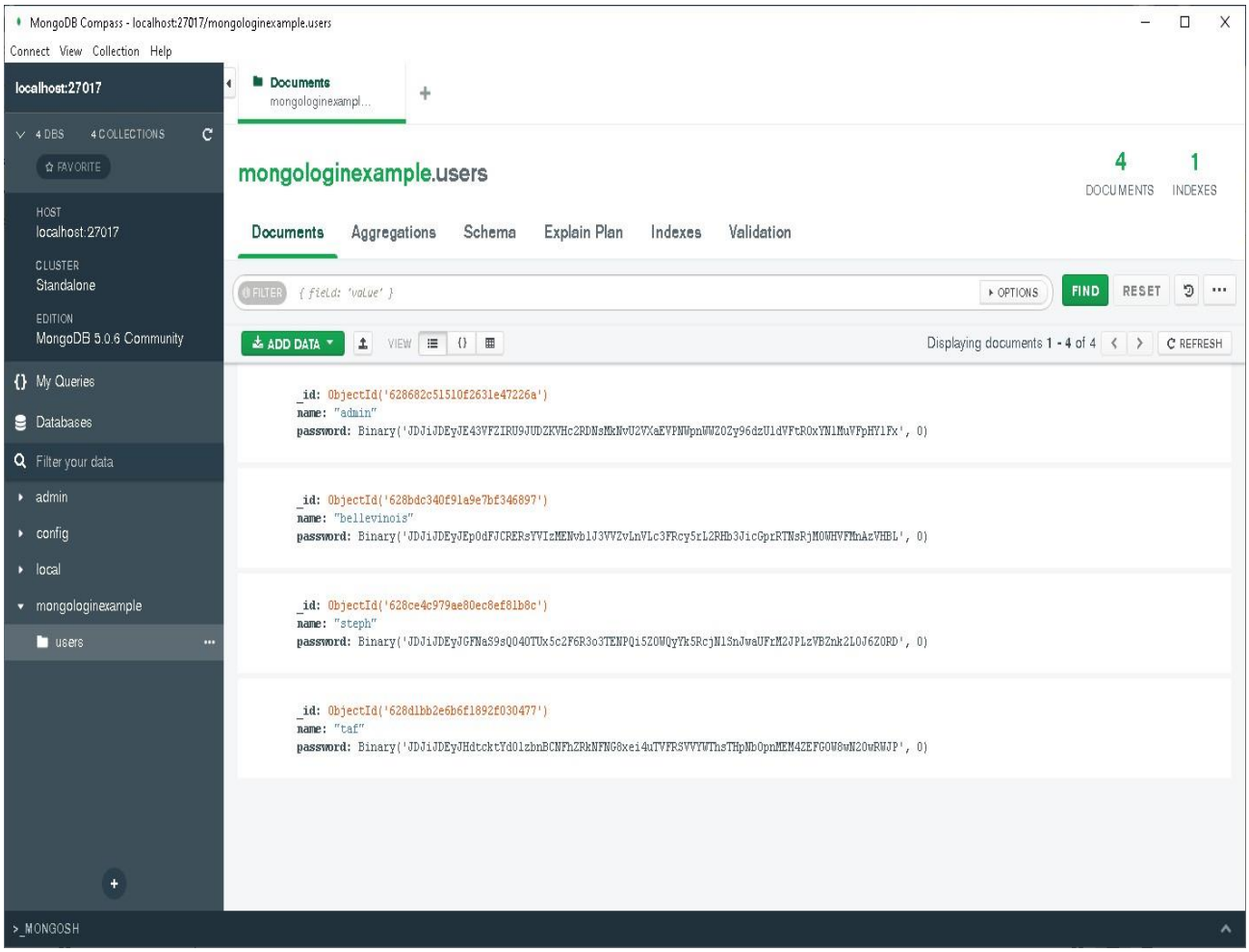


Рис. 17. База данных MongoDB

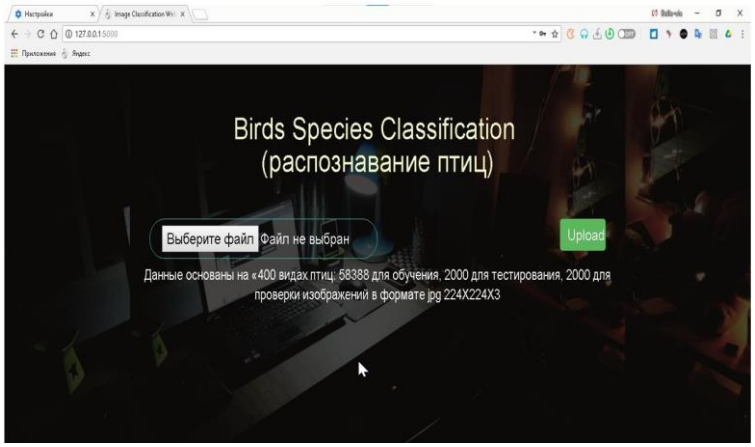


Рис. 18. Главная страница приложения

Главная страница (home) доступна после аутентификации. На этой странице можно загрузить фотографию с жесткого диска компьютера и проверить класс птица на картинке.

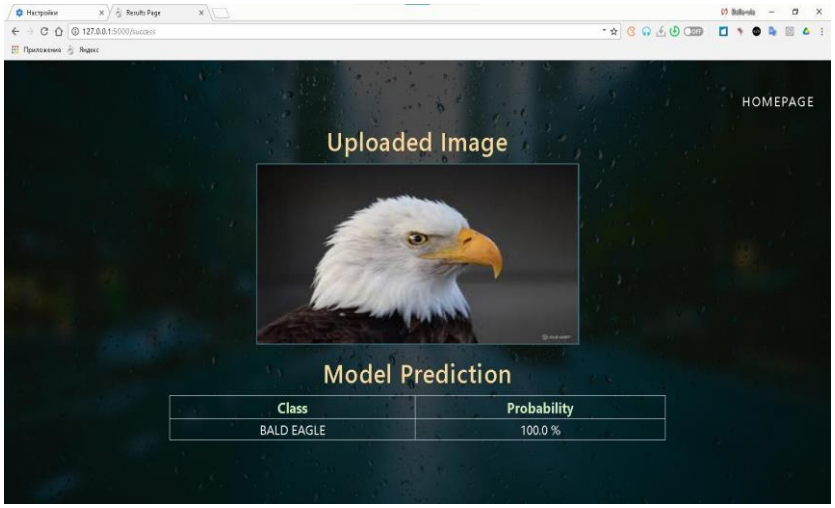


Рис. 19. Страница результата предсказания

На странице success (рисунок 19) представлен результаты предсказания с вероятностью принадлежности к определенному классу.

Заключение

Достигли конца этой статье, целью которой была описать процесс разработка и проектировка веб-приложение, способного распознавать птиц на фотографии с помощью алгоритма, основанного на нейронных сетях VGG16 модуля Keras. В процессе был разработан пользовательский интерфейс как в backend, так и в frontend с помощью фреймворка Flask. Стоит отметить, что реализованное приложение довольно простое и хорошо справляется со своей задачей, для которой оно было разработано. Более того, была разработана система аутентификации по логину и паролю для доступа к самому приложению. Важно отметить, что для получения отличных результатов на валидационные выборки, модель была переобучена на 20 эпохах.

Литература

1. Deploy Keras Models using TensorFlow Serving – TF 2.x. – Режим доступа: https://towardsdatascience.com/serving-keras- models-locally-using-tensorflow-serving-tf-2-x-8bb8474c304e. – (Дата обращения: 12.04.2022).

2. Quickstart with flask. – Режим доступа: https://flask.palletsprojects.com/en/1.1.x/quickstart/#a-minimal-application. – (Дата обращения: 12.04.2022).

3. How to Make a Web Application Using Flask in Python 3 – Режим доступа: https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3. – (Дата обращения: 12.04.2022).

УДК *917.039*

ПРОСТРАНСТВА СУММИРУЕМЫХ ФУНКЦИЙ С НОСИТЕЛЕМ НА СЕТЕПОДОБНОЙ ОБЛАСТИ

С. А. Баталова

Воронежский государственный университет

Введение

Представлены подходы формирования пространств суммируемых функций с носителем на сетеподобной области, каковыми являются области, примыкающие друг к другу своими границами, но типу графа-дерево. Элементами таких пространств являются суммируемые на сетеподобной области функции. Указанные пространства используются для анализа двухслойной и трехслойной дифференциально разностной схемы с весовым параметром в классе функций на сетеподобной области. Результаты применимы для получения достаточных условий разрешимости дифференциальной системы с распределенными весовыми параметрами на сетеподобной области.

1. Основные понятия

Сетеподобная ограниченная область $\Im \subset R^n$ с границей $\partial\Im$ состоит из областей $\Im_l$ с границами $\partial\Im_l$ ($l = \overline{1,N}$), соединенных в M узловых местах ω_j ($j = \overline{1,\mathrm{M}}$,$1 \le \mathrm{M} \le \mathrm{N}-1$): ω_j

($j = \overline{1,\mathrm{M}}$, $\le \mathrm{M} \le \mathrm{N}-1$): $\Im = \widehat{\Im} \cup \widehat{\omega}$, $\widehat{\Im} = \bigcup_{l=1}^{\mathrm{N}} \Im_l$, $\widehat{\omega} = \bigcup_{j=1}^{\mathrm{M}} \omega_j$[1, 2]. В каждом узловом месте ω_j ($j = \overline{1,\mathrm{M}}$) фиксированное число подобластей $\Im_j$ имеют общие границы, образующие поверхность их примыкания S_j (meas $S_j > 0$). Эта поверхность связывает между собой примыкающие к ней $1 + m_j$ области $\Im_{l_0}$ и $\Im_{l_S}$ ($s = \overline{1,m_j}$): $S_j = \bigcup_{s=1}^{m_j} S_{j_s}$(meas $S_{j_s} > 0$), $S_j \subset \partial\Im_{l_0}$, $S_{j_s} \subset \partial\Im_{l_s}(s = \overline{1,m_j})$. Узловое место ω_j ($j = \overline{1,\mathrm{M}}$) определяется своею поверхностью примыкания S_j, для которой каждая поверхность S_{j_s} ($s = \overline{1,m_j}$) также является поверхностью примыкания $\Im_{l_s}$ к $\Im_{l_0}$. При этом граница области $\Im$ не содержит S_j ($j = \overline{1,\mathrm{M}}$): $\partial\Im = \bigcup_{\mathrm{k}=1}^{\mathrm{N}} \partial\Im_k \setminus \bigcup_{j=1}^{\mathrm{M}} S_j$. Структура области $\Im$ совпадает с геометрией графа-дерево с внутренними узлами (вершинами) ω [3, 4]; любая подобласть области $\Im$ также имеет аналогичную с $\Im$ структуру (см. Рис.).

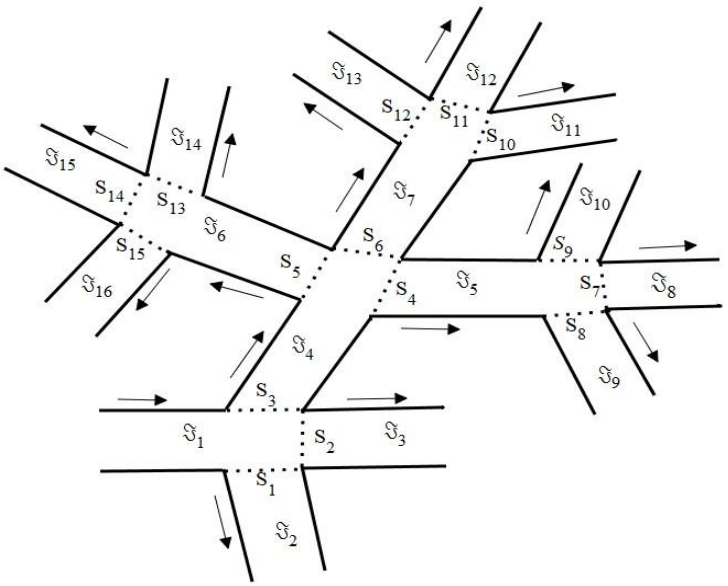


Рис. 1. Сетеподобная область $\Im$

Используются классические пространства Лебега и Соболева. Пусть $L_2(\Omega)$ $(\Omega \subset R^n)$ – гильбертово пространство действительных измеримых по Лебегу функций $u(x)$, $x = (x_1, x_2, \dots, x_n)$, скалярное произведение и норма в $L_2(\Omega)$ определены равенствами $(u, v)_\Omega = \int_\Omega u(x)v(x)dx$ и $||$ и $||_\Omega = \sqrt{(u,u)}$. Далее, $W_2^1(\Omega)$ – соболевское пространство элементов $u(x) \in L_2(\Omega)$, для которых $u_{x_k}(x) \in L_2(\Omega)$, $\mathrm{K} = \overline{1,n}$. Скалярное произведение и норма в $W_2^1(\Omega)$ определены соотношениями

$$(u,v)_\Omega^{(1)} = \int_\Omega (uv + \sum_{\kappa=1}^{n} u_{x_\kappa} v_{x_\kappa})dx = \int_\Omega \left(u(x)v(x) + \sum_{\kappa=1}^{n} \frac{\partial u(x)}{\partial x_\kappa} \frac{\partial v(x)}{\partial x_\kappa} \right) dx, \tag{1}$$

$$|| u ||_\Omega^{(1)} = \sqrt{(u,u)_\Omega^{(1)}} \tag{2}$$

Применительно к сетеподобной области $\Im$ имеем $\int_\Im u(x)dx = \sum_{k=1}^{\mathrm{N}} \int_{\Im_k} u(x)dx$ и соотношения (1), (2) принимают следующий вид:

$$\mathrm{P}u\,\mathrm{P}_\Im = (\sum_{k=1}^{N} \int_{\Im_k} u^2(x)dx)^{1/2}, \tag{3}$$

$$(u,v)_\Im^{(1)} = \sum_{k=1}^{N} (u,v)_{\Im_k}^{(1)} = \sum_{k=1}^{N} \int_{\Im_k} \left(u(x)v(x) + \sum_{\kappa=1}^{n} \frac{\partial u(x)}{\partial x_\kappa} \frac{\partial v(x)}{\partial x_\kappa} \right) dx, \tag{4}$$

$$\mathrm{P}u\,\mathrm{P}_\Im^{(1)} = (\sum_{k=1}^{N} (u,u)_{\Im_k}^{(1)})^{1/2}. \tag{5}$$

Обозначим через $C(\overline{\Im})$ множество непрерывных в $\overline{\Im}$ функций $u(x)$, через $C^1(\overline{\Im_k})$ $(k =$

$\overline{1,\mathrm{N}}$) множества функций из $C(\overline{\mathfrak{I}})$, для которых при каждом фиксированном k в $\overline{\mathfrak{I}_k}$ существуют непрерывные производные $u_{x_1}(x)$, $u_{x_2}(x)$,..., $u_{x_n}(x)$, а через $C^1(\overline{\mathfrak{I}})$ множество функций со скалярным произведением и нормой, определяемыми формулами (3), (4) и (5), соответственно.

Пусть $\widetilde{C^1}(\overline{\mathfrak{I}})$ – множество функций $u(x) \in C^1(\overline{\mathfrak{I}})$, для которых имеет место соотношения (условия примыкания)

$$\int_{S_j} a(x)_{S_j} \frac{\partial u(x)_{S_j}}{\partial n_j} ds + \sum_{i=1}^{m_j} \int_{S_{ji}} a(x)_{S_{ji}} \frac{u(x)_{S_{ji}}}{\partial n_{ji}} ds = 0, \quad x \in S_{ji}, i = \overline{1, m_j}, \tag{3}$$

на поверхностях S_j, S_{j_i} $(i = \overline{1, m_j})$ всех узловых мест ω_j, $j = \overline{1,\mathrm{M}}$. Здесь $a(x) \in L_2(\overline{\mathfrak{I}})$ и $a(x)_{s_j}$, $u(x)_{s_j}$, $a(x)_{s_{ji}}$, $u(x)_{s_{ji}}$– сужения функций $a(x) \in L_2(\mathfrak{I})$, u(x) на S_j и S_{ji}, векторы n_j и n_{ji} – внешние нормали к S_j и S_{ji}, соответственно, $i = \overline{1, m_j}$, $j = \overline{1,\mathrm{M}}$.

Определение 1. *Замыкание* $\widetilde{C^1}(\overline{\mathfrak{I}})$ *в норме, представленной соотношением* (5), *назовем пространством* $\widetilde{W^1}(\mathfrak{I})$, $||\cdot|\ \ |_{\widetilde{W^1}(\mathfrak{I})} = ||\cdot|\ \ |_{W_2^1(\mathfrak{I})} := ||\cdot||_{\mathfrak{I}}^1$.

Представление $C^1(\overline{\mathfrak{I}})$ определяет особенность пространства $\widetilde{W^1}(\mathfrak{I})$: если $u(x) \in \widetilde{W^1}(\mathfrak{I})$, то сужение $u(x)_{\mathfrak{I}_k} \in W^1(\mathfrak{I}_k)$, $k = \overline{1,\mathrm{N}}$. Из $\mathfrak{I}_k \subset \mathfrak{I}$ $(k = \overline{1,\mathrm{N}})$ и существования обобщенной производной $u_x(x)$ в области $\mathfrak{I}$ следует существование $u_x(x)_{\mathfrak{I}_k}$ в $\mathfrak{I}_k$.

Пусть далее $\widetilde{C_0^1}(\mathfrak{I})$ – множество элементов из $\widetilde{C^1}(\overline{\mathfrak{I}})$ с компактным носителем в области $\mathfrak{I}$, таким образом, элементы множества $\widetilde{C_0^1}(\mathfrak{I})$ равны нулю вблизи границы $\partial\mathfrak{I}$.

Определение 2. *Замыкание* $\widetilde{C_0^1}(\mathfrak{I})$ *в норме, представленной соотношением* (5) *назовем пространством* $\widetilde{W_0^1}(\mathfrak{I})$.

Замечание 1. Для элементов $u(x)$ пространства $\widetilde{W_0^1}(\mathfrak{I})$ можно ввести другое скалярное произведение и норму

$$[u,v]_{\mathfrak{I}}^{(1)} = \sum_{k=1}^{N} [u,v]_{\mathfrak{I}_k}^{(1)} = \sum_{k=1}^{N} \int_{\mathfrak{I}_k} \sum_{\kappa=1}^{n} \frac{\partial u(x)}{\partial x_\kappa} \frac{\partial v(x)}{\partial x_\kappa} dx,$$

порождающее норму

$$\mathrm{P}u\,\mathrm{P}_{[]}^{(1)} = \sqrt{[u,u]_{\mathfrak{I}}^{(1)}}.$$

Эквивалентность норм $||u||^{(1)}$ и $||u||_{[]}^{(1)}$ устанавливается с помощью аналога неравенства Пуанкаре-Фридрихса $\int_{\mathfrak{I}} u^2(x)dx \le c \int_{\mathfrak{I}} u_x^2(x)dx$ (C – постоянная, зависящая только от области $\mathfrak{I}$), доказательство которого аналогично представленному в [5, с. 98].

Учитывая $\widetilde{W_0^1}(\mathfrak{I}) \subset \widetilde{W^1}(\mathfrak{I})$, из определения 2 следует, что $\widetilde{W_0^1}(\mathfrak{I})$– сепарабельное банахово пространство. В силу замкнутости подпространства гильбертова пространства получаем: из слабой сходимости последовательности в $\widetilde{W_0^1}(\mathfrak{I})$ следует, что ее предельный элемент принадлежит этому пространству.

Отметим, что $\widetilde{W_0^1}(\mathfrak{I})$ используется для анализа краевых задач с условиями Дирихле, $\widetilde{W_0^1}(\mathfrak{I})$– для изучения краевых задач с общими краевыми условиями.

2. Использование введенных пространств для анализа операторно-разностных и дифференциальных систем

В пространстве $\widetilde{W_0^1}(\mathfrak{I})$ (или) $\widetilde{W^1}(\mathfrak{I})$ рассматриваются операторно-разностные схемы с весовыми параметрами σ_1 и σ_2 (σ_1 , σ_2 – вещественные числа), от выбора которых зависит устойчивость и точность схем.

На отрезке $[0,T]$ введем равномерную сетку с шагом $\tau = T/K$: $\omega_\tau = \{t_k = k\tau, k = 0,1,\dots,K\}$. Для функций $y(k) := y(x;k)$, k = 0 ,1 , … ,K , примем следующие обозначения, учитывая границы изменения индекса k [6, с. 346]:

$$
\begin{aligned}
&y = y(k), \quad \hat{y} = y(k+1), \quad \check{y} = y(k-1),\\
&y_t(0) = \frac{1}{\tau}(y(1)-y(0)), \quad y_t = \frac{1}{\tau}(\hat{y}-y), \quad y_{\bar{t}} = \frac{1}{\tau}(y-\check{y}), y_{\overset{\circ}{t}} = \frac{1}{2\tau}(\hat{y}-\check{y}),\\
&y_{\bar{t}t} = \frac{1}{\tau^2}(\hat{y}-2y+\check{y}), \quad y^{(\sigma_1,\sigma_2)} = \sigma_1\hat{y}+(1-\sigma_1-\sigma_2)y+\sigma_2\check{y}.
\end{aligned}
\tag{4}
$$

Из (7) вытекают следующих соотношения:

$$
\begin{aligned}
&y_t = y_{\overset{\circ}{t}} + \frac{\tau}{2}y_{\bar{t}t}, \quad y_{\bar{t}} = y_{\overset{\circ}{t}} - \frac{\tau}{2}y_{\bar{t}t},\\
&y = \frac{1}{2}(\hat{y}+\check{y}) - \frac{1}{2}(\hat{y}-2y+\check{y}) = \frac{1}{2}(\hat{y}+\check{y}) - \frac{\tau^2}{2}y_{\bar{t}t},\\
&\hat{y} = y + \frac{1}{2}(\hat{y}-\check{y}) + \frac{1}{2}(\hat{y}-2y+\check{y}) = y + \tau y_{\overset{\circ}{t}} + \frac{\tau^2}{2}y_{\bar{t}t},\\
&\check{y} = y - \frac{1}{2}(\hat{y}-\check{y}) + \frac{1}{2}(\hat{y}-2y+\check{y}) = y - \tau y_{\overset{\circ}{t}} + \frac{\tau^2}{2}y_{\bar{t}t},\\
&y^{(\sigma_1,\sigma_2)} = y + (\sigma_1-\sigma_2)\tau y_{\overset{\circ}{t}} + (\sigma_1+\sigma_2)\frac{\tau^2}{2}y_{\bar{t}t}.
\end{aligned}
\tag{5}
$$

В пространстве $\widetilde{W_0^1}(\mathfrak{I})$ введем оператор

$$
Au = -\frac{\partial}{\partial x_к}\left(a_{кl}(x)\frac{\partial u}{\partial x_l}\right) + b(x)u, \qquad \frac{\partial}{\partial x_к}\left(a_{кl}(x)\frac{\partial u}{\partial x_l}\right) := \sum_{к,l=1}^{n}\frac{\partial}{\partial x_к}\left(a_{кl}(x)\frac{\partial u}{\partial x_l}\right),
$$

здесь

$$a_{\kappa l}(x) = a_{l\kappa}(x), \quad |b(x)| \le \beta, \qquad x \in \Im,$$

$$a_* \xi^2 \le \sum_{\kappa,l=1}^{n} a_{\kappa l}(x)\xi_\kappa \xi_l \le a^* \xi^2, \quad \xi^2 = \sum_{\kappa=1}^{n} \xi_\kappa^2,$$

с фиксированными положительными постоянными a_*, a^*, β и произвольными параметрами $\xi_1, \xi_2, \dots, \xi_n$. Рассмотрим

а) двухслойную операторно-разностные систему с весами σ_1, σ_2для параболической системы уравнений

$$y_t + Ay^{(\sigma_1,\sigma_2)} = f(k), \qquad k = \overline{1, K-1}, \qquad y(0) = \varphi_0(x), \qquad (9_1)$$

б) трехслойную операторно-разностные систему с весами σ_1, σ_2для гиперболической системы уравнений

$$y_{\bar{t}t} + Ay^{(\sigma_1,\sigma_2)} = f(k), \qquad k = \overline{1, K-1}, \qquad y(0) = \varphi_0(x), \qquad y_t(0) = \varphi_1(x), \qquad (9_2)$$

Где $f(k) \coloneqq f(x;k)$, $k = \overline{1, K-1}$. Для каждого фиксированного k ($k = \overline{1, K-1}$) функция $y(k+1) \in \widetilde{W_0^1}(\Im)$ является решением (9_1) (или (9_2)) и удовлетворяет краевому условию

$$y(k+1)|_{x \in \partial\Gamma} = 0, \qquad (6)$$

при этом $y(1) = y(0) - \tau y_t(0) = \varphi_0(x) - \tau\varphi_1(x)$ и предполагаются выполненными условия

$$\varphi_0(x), \varphi_1(x) \in W_0^1(\Im), \quad f(k) \coloneqq f(x;k) \in L_2(\Im) \quad (k = \overline{1, K-1}).$$

Определение 3. *Совокупность функций* $y(k) \in \widetilde{W_0^1}(\Im)$, $k = 2, \dots, K$, *называется слабым решением системы* (9_2), (10), *если удовлетворяются тождества*

$$\int_\Im y_{\bar{t}t}\eta(x)dx + \ell(y^{(\sigma_1,\sigma_2)}, \eta) = \int_\Im f(k)\eta(x)dx \qquad \forall \eta(x) \in W_0^1(\Im), \quad k = \overline{1, K-1},$$

$$y(0) = \varphi_0(x), \quad y_t(0) = \varphi_1(x),$$

для каждого $y(k)$; *здесь*

$$\ell(y^{(\sigma_1,\sigma_2)}, \eta) = \int_\Im \left(\sum_{\kappa,l=1}^{n} a_{\kappa l}(x) \frac{\partial y^{(\sigma_1,\sigma_2)}}{\partial x_l} \frac{\partial \eta(x)}{\partial x_\kappa} + b(x) y^{(\sigma_1,\sigma_2)} \eta(x) \right) dx.$$

Замечание 2. Аналогичное определение вводится для (9_1).

Замечание 3. При каждом фиксированном k ($k = 1, 2, \dots, K-1$) соотношения (9_2), (10) в $\widetilde{W_0^1}(\Im)$ описывает краевую задачу относительно $y(k+1)$ ($y(k+1) = \hat{y}$).

Нетрудно заметить, что, учитывая аппроксимации производных $\frac{\partial u}{\partial t}$, $\frac{\partial^2 u}{\partial t^2}$разностными соотношениями u_t, $u_{\bar{t}t}$(см. (7), (8)), операторно-разностные системы (9_1) и (9_2) определены для дифференциальных систем

$$\frac{\partial u}{\partial t} + Au^{(\sigma_1,\sigma_2)} = f(x,t), \quad u(x,0) = \varphi_0(x),$$

$$\frac{\partial^2 u}{\partial t^2} + Au^{(\sigma_1,\sigma_2)} = f(x,t), \quad y(x,0) = \varphi_0(x), \quad \frac{\partial u}{\partial t}(x,0) = \varphi_1(x),$$

соответственно; здесь $f(k) \coloneqq f(x;k)$ и $f(x,t)$ связаны соотношениями

$$f(k) \coloneqq f(x;k) = \frac{1}{\tau}\int_{(k-1)\tau}^{k\tau} f(x,t)dt, \quad k = 1,2,...,K.$$

Откуда следует прямая связь вопроса разрешимости дифференциальных систем с анализом априорных оценок решений соответствующих операторно-разностных систем (см. аналогичный подход в работе [7]).

Заключение

Представленные результаты раздела 2 определяют условие слабой разрешимости дифференциально параболических и гиперболических систем, с пространственными переменными, принадлежащими n-мерному евклидовому пространству. Эти результаты можно использовать в задачах оптимизации, возникающих при моделировании сетеподобных процессов переноса сплошных сред.

Научный руководитель доцент, доктор физ.-мат.наук, профессор кафедры уравнений в частных производных и теории вероятностей ВГУ, Провоторов Вячеслав Васильевич.

Литература

1. Provotorov V V and Provotorova E N 2017 Optimal control of the linearized Navier-Stokes system in a netlike domain // Vestnik of Saint Petersburg University. Series 10. Applied Mathematics. Computer Science. Control Processes, 2017, vol. 13. no. 4. pp. 431-443.

2. Borisoglebskaya L.N.,. Provotorov V.V, Sergeev S.M. and Kosinov E.S. Mathematical aspects of optimal control of transference processes in spatial networks. IOP Conference Series: Materials Science and Engineering (2019), Volume 573.

3. Баталова С. А. Дифференциально-разностная система в классе суммируемых на графе функций // Современные методы прикладной математики, теории управления и компьютерных технологий (ПМТУКТ-2021) : сборник трудов Всероссийской научной конференции. – Воронеж, 2021. – С. 12-13.

4. Баталова С. А. Разрешимость дифференциально-разностной системы параболического типа с распределенными параметрами на графе // Международная научная конференция «Воронежская зимняя математическая школа С. Г. Крейна – 2022» – Воронеж, 2022. – С. 20-25.

5. Провоторов В. В., Волкова А. С. Начально-краевые задачи с распределенными параметрами на графе. Воронеж: Научная книга. 2014. 188 с.

6. Самарский А. А. Теория разностных схем, Наука, М., 1977, 656 с.

7. Zhabko A. P., Provotorov V. V., Shindyapin A. I. Optimal control of a differential-difference parabolic system with distributed parameters on the graph // Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. 2021. vol. 17. iss. 4. pp. 433-448.

УДК *004.4’2*

ИССЛЕДОВАНИЕ ПОДХОДОВ К РАЗРАБОТКЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА ПРИМЕРЕ СИСТЕМЫ УПРАВЛЕНИЯ ГРУЗОПЕРЕВОЗКАМИ

О. А. Безрукова

Воронежский государственный университет

Введение

В современном мире сложно найти человека, у которого не было бы смартфона или планшета. Компактные, многофункциональные устройства достаточно давно являются неотъемлемыми помощниками в повседневной жизни человека: с помощью них можно как сделать звонок или написать сообщение, так и, например, заказать такси или оформить заказ на покупку чего-либо.

Каждый год появляются всё новые технологии, которые позволяют относительно быстро создавать мобильные приложения. Это различные фреймворки, библиотеки, которые значительно упрощают процесс разработки. На сегодняшний день есть перечень наиболее часто используемых технологий для разработки мобильных приложений. Среди фреймворков – это Flutter, lonic, React Native, Xamarin, а среди библиотек – Firebase, Dagger, Retrofit.

Любое приложение нацелено на то, чтобы облегчить жизнь человека. Часто, когда люди переезжают или покупают какие-либо крупногабаритные вещи, например, стиральную машинку или телевизор и не имеют возможности перевезти покупку самостоятельно, то они заказывают доставку. Раньше это можно было сделать в магазине или по объявлению. На сегодняшний же день также существуют сервисы, которые позволяют, подобно такси, сделать заказ на доставку крупного груза. Удобнее всего это делать через мобильное приложение, которое может позволить не только заказать грузовое такси, но и в режиме реального времени видеть как водитель, который принял этот заказ, добирается до заказчика.

Ранее, в сборнике «Актуальные проблемы прикладной математики, информатики и механики» в статье «Исследование подходов к разработке мобильных приложений на примере системы управления грузоперевозками» были рассмотрены основные кроссплатформенные фреймворки для разработки мобильных приложений, были определены критерии сравнения программ, разработанных с использованием Flutter и React Native, а также описаны ключевые этапы разработки приложения с использованием фреймворка Flutter.

Таким образом, целью данной работы является разработка мобильного приложения с использованием фреймворка React Native, а также дополнительных инструментов, таких как Firebase и Mapbox.

1. Анализ задачи

Необходимо разработать приложение, которое позволяло бы заказать доставку, например, крупногабаритных грузов (сыпучие строительные материалы) или доставку мебели, какой-либо бытовой техники.

Исходя из описания, можно выделить две основные роли в приложении – заказчик и водитель. Также отдельно можно рассматривать обычного пользователя, который не авторизовался в системе или вовсе не имеет аккаунта и администратора.

После установки приложения пользователю предлагается две опции – войти в свой профиль, если он уже существует или зарегистрировать новый.

После того, как пользователь зашел в систему, ему предлагается набор действий, представленный на рис. 1.

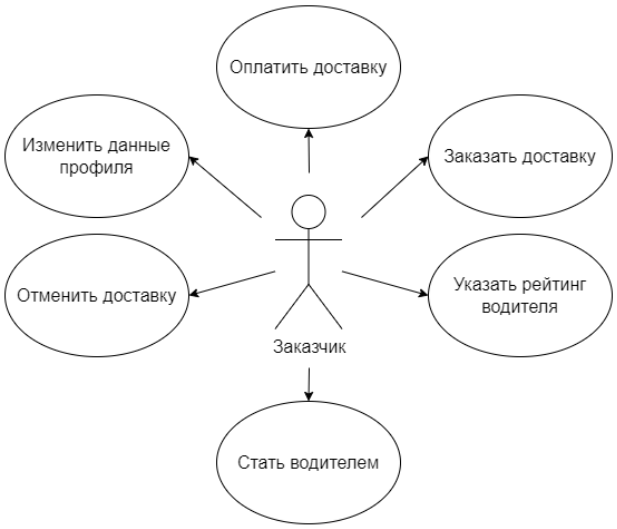


Рис. 1. Диаграмма вариантов использования для роли «Заказчик»

Для того чтобы стать водителем для начала необходимо быть обычным пользователем приложения (заказчиком). В личном кабинете для всех пользователей доступна кнопка «Стать водителем», при нажатии на которую появляется форма, которую необходимо заполнить. В ней запрашивается информация о машине (марка, номер), категории (выбирается в соответствии с диапазоном максимально допустимых грузов), а также фото автомобиля.

Когда обычный пользователь становится водителем, то ему становится доступным набор действий, представленный на рис. 2.

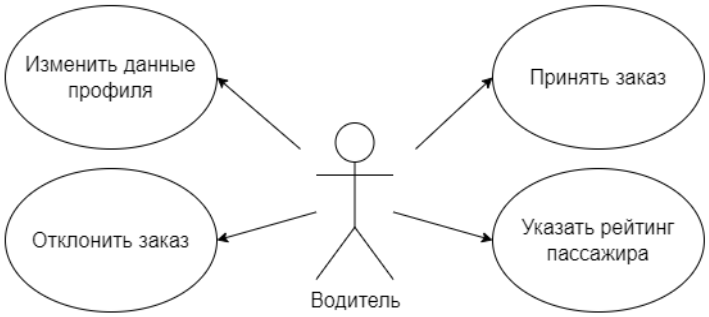


Рис. 2. Диаграмма вариантов использования для роли «Водитель»

Также в системе есть роль администратора, который просматривает заявки пользователей на получение статуса водителя. Если администратора устраивает отправленная информация о машине, ее состояние, то пользователю выдается роль водителя, в противном случае отправляется отказ.

Взаимодействие пользователя (заказчика) и водителя происходит следующим образом. Пользователь заказывает доставку, для этого он заполняет форму, в которой необходимо указать адрес, откуда забирать груз, адрес куда доставить и примерную весовую категорию для корректного подбора машины. После этого система предложит заказ первым пяти водителям, подходящим под подобранную категорию. Если все пять водителей откажутся от заказа, то

далее будет предложено следующим по списку водителям. В случае если все откажутся, то будет отображено соответствующее сообщение пользователю.

Когда заказ принят водителем, то пользователю показывается его местоположение, а для водителя стоится маршрут, до точки отправления. В момент, когда водитель добрался до нужного места, ему необходимо нажать кнопку, которая перестроит маршрут уже от точки отправления до конечной цели.

Пользователю доступно два вида оплаты: картой или наличными. Карту можно привязать в личном кабинете.

2. Используемые инструменты

В разделе рассматриваются основные инструменты, которые используются для разработки приложения.

2.1. Expo CLI и Expo Go

Expo CLI[2] – это приложение командной строки, которое является основным интерфейсом между разработчиком и инструментами Expo. Expo CLI также имеет веб-интерфейс, который можно запустить браузере при запуске проекта (можно использовать графический интерфейс вместо интерфейса командной строки).

Expo Go – это приложение для IOS и Android платформ, позволяющее открывать разработанные, с использованием Expo CLI, программы на мобильном устройстве.

2.2. React Native

React Native[3] – это кроссплатформенный фреймворк с открытым исходным кодом для разработки нативных мобильных и настольных приложений на JavaScript и TypeScript.

В работе используется как основной инструмент разработки.

2.3. Firebase

Firebase[4] – это платформа для разработки мобильных приложений от компании Google, в которой есть самые современные функции для разработки, перекомпоновки и улучшения приложений. Это, по своей сути, набор инструментов, которые разработчики могут использовать, создавая и изменяя приложения в зависимости от своей потребности.

В работе используется в качестве хранилища данных – базы данных и для аутентификации. Также при разработке, для отслеживания показателей, использовались мониторинговые функции, предоставляемые Firebase. Например, App Checker, Performance.

2.4. Mapbox

Mapbox[5] – это сервис, который позволяет создавать собственные карты, и интегрировать их в приложение. Основным преимуществом по сравнению с Google Maps является то, что базовые функции этого сервиса бесплатные.

Для успешного выполнения работы необходимо, чтобы были доступны следующие возможности:

1) отображение карты;
2) построение маршрута по указанным точкам (долгота, широта для исходной и конечной точек);
3) навигация;

4) определение местоположения с использованием геолокации;
5) подсчет времени, необходимого для завершения поездки из пункта А в пункт Б.

Данный сервис предоставляет все вышеперечисленные возможности.

В работе используется для определения геолокации по датчикам мобильного устройства или по адресу, указанному пользователем, для построения маршрута, навигации и подсчета времени поездки.

3. Разработка

В разделе рассматриваются основные этапы создания мобильного приложения с использованием фреймворка React Native.

3.1. Создание проекта

Для того чтобы создать проект, с использованием Expo, необходимо выполнить следующие шаги:

1) выполнить команду npx create-expo-app ${название_приложения};
2) после успешного выполнения необходимо открыть проект, например, в Android Studio или в Visual Studio Code;
3) запустить приложение можно из файла package.json (start) или с помощью команды npx expo start.

В результате выполнения указанных шагов на экране появляется QR код (рис. 3), который можно отсканировать камерой IOS или Android устройства (через Expo Go) и созданное приложение будет установлено на телефон.

Рис. 3. QR код, сгенерированный при запуске приложения

3.2. Интеграция с Firebase

Среда разработки Android Studio имеет встроенные инструменты, которые позволяют быстро настроить интеграцию приложения с Firebase. Для того чтобы это сделать необходимо в IDE перейти во вкладку Tools и выбрать Firebase, после чего откроется окно ассистента, где можно выбрать интересующие функции.

Как было написано ранее, этот инструмент используется для добавления аутентификации в приложение и в качестве базы данных.

Чтобы подключить возможность аутентификации, в ассистенте Firebase в Android Studio необходимо выбрать соответствующий пункт. Далее, при следовании указанным шагам, в проект будет добавлен файл google-services.json, который содержит необходимую конфигурацию для интеграции. После этого в консоле Firebase следует выбрать метод входа (в работе использовался вход с помощью email и пароля). После выполненных действий при попытке зайти в приложение с использованием несуществующего аккаунта, со стороны Firebase будет возвращена ошибка, говорящая о том, что такого пользователя не существует.

Для подключения к приложению базы данных – Firestore Database и хранилища – Storage – также можно воспользоваться инструкцией, предлагающейся в ассистенте Firebase.

На рис. 4. представлена коллекция «order».

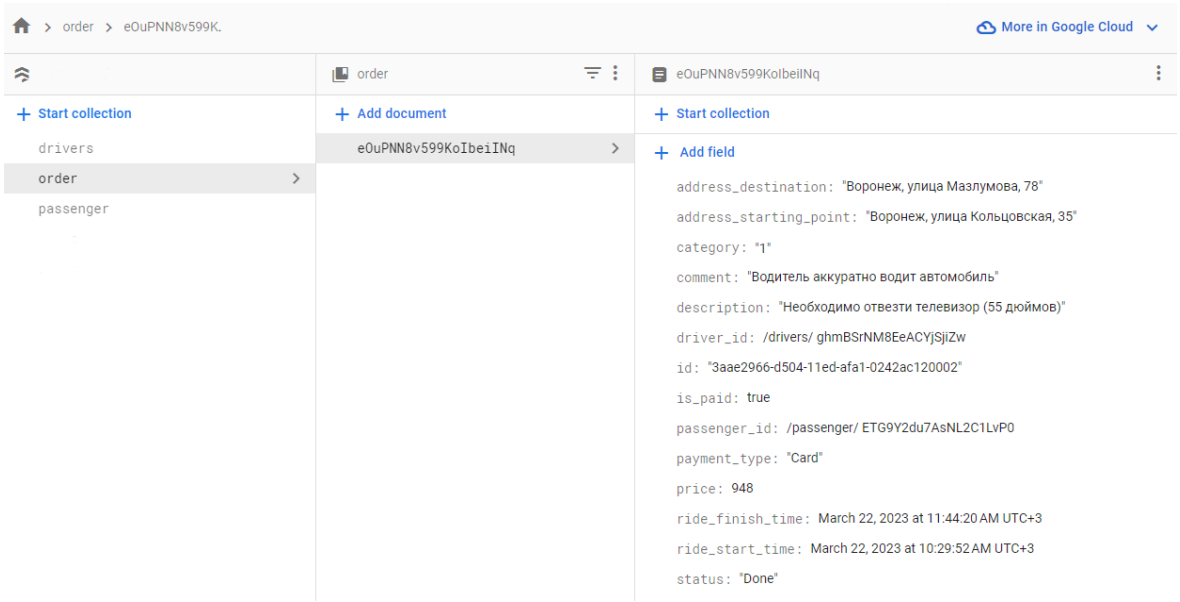


Рис. 4. Пример записи в коллекции «order» в Firestore Database

База данных состоит из трех основных коллекций – водитель, заказ, пассажир. Водитель и пассажир имеют поля, содержащие информацию о человеке – имя, фамилию, дату рождения, номер телефона, а также поля, специфичные для приложения – рейтинг, фотография, а для водителя еще дополнительно модель машины и ее государственный номер.

В коллекции «Заказ» указываются ссылки на водителя и пассажира (один ко многим), статус заказа (открыт, в процессе, выполнен), цена и информация об оплате (оплачено или нет, способ оплаты), также пассажир может оставить комментарий по окончании поездки. При создании заказа указывается информация об адресе отправления и адресе окончания, можно добавить описание (например, указать, что хотят отвезти) и указать категорию (1 – до 200кг, 2 – до 700кг, 3 – свыше 700кг). Время начала выполнения и окончания фиксируется приложением.

На рис. 5. представлена схема базы данных.

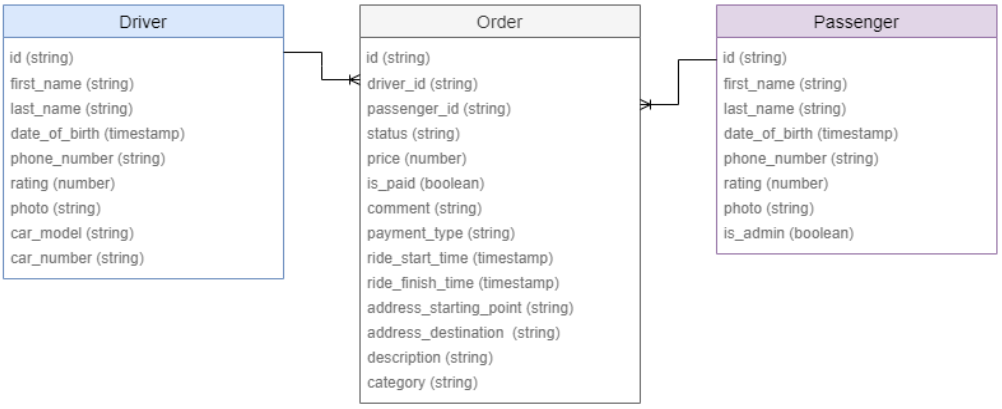


Рис. 5. Схема базы данных

Для того чтобы из кода приложения была возможность обращаться к базе данных, необходимо добавить импорт firestore из @react-native-firebase/firestore. После этого можно работать с данными. Например, на рис. 6. представлен фрагмент кода, с помощью которого можно обратиться к документам в коллекции.

```
firestore().collection(${название_коллекции}).get().then(querySnapshot => {
    querySnapshot.forEach(documentSnapshot => {
      // Логика для работы с документами коллекции
    });
  });
```

Рис. 6. Пример получения документов из коллекции

3.3. Интеграция с Mapbox

Для работы с картами в приложении используются нативные элементы, то есть при загрузке приложения загружаются карты, которые используются в системе по умолчанию. Стоит отметить, что при работе с Android приложением используются Google Maps, а для их использования требуется ключ, который можно создать в Google консоле. С устройством на платформе IOS такой проблемы не возникает, так как стандартное приложение не требует каких-либо дополнительных действий для работы.

Как было сказано ранее, в приложении сторонний сервис Mapbox используется для навигации, построения маршрутов, определения геолокации, подсчета времени поездки. Здесь, как и во многих сервисах, необходимо иметь аккаунт, чтобы использовать уникальный ключ доступа при выполнении запросов.

Когда ключ доступа есть, то можно по API обращаться к таким сервисам Mapbox, как Geocoding API (часть Search API), Directions API, Matrix API (часть Navigation API). Для каждого API добавляется отдельный .ts файл для работы. Например, для работы с Directions API используется DirectionsService.ts файл. В нем указывается uri с необходимыми параметрами – координаты начальной точки и конечной (долгота и широта), также в запросе передается ключ и другие вспомогательные параметры. Используемая конечная точка /driving-traffic позволяет учитывать трафик на дорогах, что позволяет по максимуму избегать пробок на дорогах. На рис. 7. представлен пример построения маршрута от главного корпуса ВГУ до кинотеатра Пролетарий.

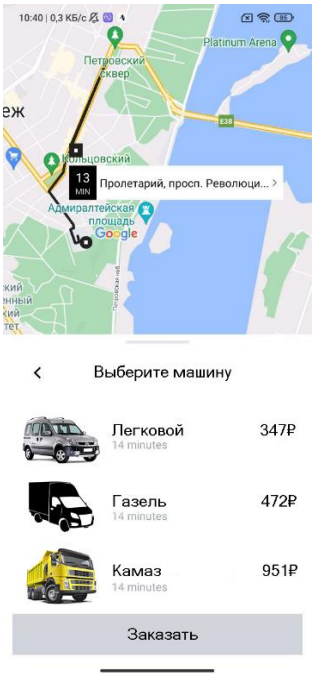


Рис. 7. Пример построение маршрута

4. Мониторинг приложения

Одной из важных частей любой разработки является мониторинг приложения. С его помощью можно отслеживать скорость, количество успешных и неуспешных запросов, нагрузку на процессор, память и многое другое. Для мобильных приложений, одними из инструментов отслеживания состояния устройства и приложения, могут использоваться Firebase и сам девайс. Основные показатели, такие как используемая память, CPU (Central Processing Unit), FPS (Frames Per Second), температура могут быть измерены устройством. Для этого необходимо включить режим разработчика и активировать Power Monitor функцию.

Для того чтобы отслеживать продолжительность выполнения запросов, скорость загрузки приложения, количество пользователей (например, для проведения нагрузочного тестирования, чтобы знать, какое количество людей может пользоваться приложением одновременно) и другие показатели внутри приложения, можно воспользоваться инструментами Firebase, а именно: App Check, Performance, Analytics Dashboard. На рис. 8. представлены данные по запуску приложения.

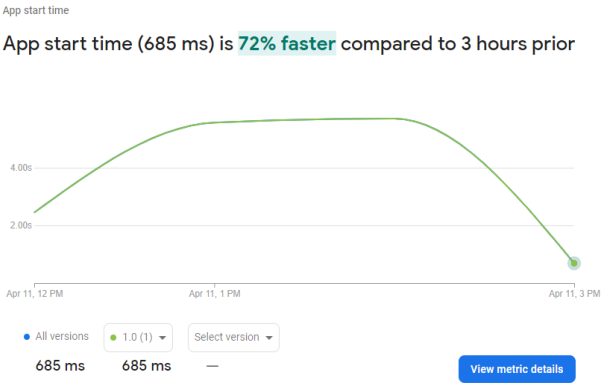


Рис. 8. Данные о скорости запуска приложения

Для наглядности была добавлена задержка загрузки, чтобы показать, как меняется

график.

Заключение

Таким образом, в процессе работы было создано приложение, которое позволяет делать заказы на доставку каких-либо крупных грузов, строить маршруты для водителей и отображать его местонахождение на карте для пассажира, рассчитывать время поездки. Разработана схема базы данных, рассмотрены основные функции и методы интеграции в приложение сторонних сервисов Firebase и Mapbox. Также описаны некоторые инструменты, которые можно использовать для мониторинга приложения.

Литература

1. Create React Native app using Expo CLI or React Native CLI. – Режим доступа: https://blog.codemagic.io/step-by-step-guide-to-kick-off-your-first-react-native-project. – (Дата обращения: 14.02.2023).

2. React Native documentation. – Режим доступа: https://reactnative.dev. – (Дата обращения: 15.02.2023).

3. Firebase documentation. – Режим доступа: https://firebase.google.com. – (Дата обращения: 22.03.2023).

4. Mapbox documentation. – Режим доступа: https://www.mapbox.com. – (Дата обращения: 29.03.2023).

5. React Native Firebase. – Режим доступа: https://rnfirebase.io. – (Дата обращения: 29.03.2023).

УДК 004.415.532.2

ПОДХОД К ТЕСТИРОВАНИЮ НА ОСНОВЕ СВОЙСТВ (НА ПРИМЕРЕ СОБСТВЕННОЙ РЕАЛИЗАЦИИ КЛАССА MAP)

А. Д. Бирюлев, К. С. Некрасов

Воронежский государственный университет

Введение

Тестирование в настоящее время является неотъемлемой частью разработки программного обеспечения любой сложности, оно позволяет выявлять ошибки на разных стадиях реализации программы. Тестирование можно осуществлять как вручную, так и автоматически.

Автоматическое тестирование позволяет каждый раз запускать одни и те же тесты по заданному сценарию. Такие тесты разрабатываются отдельно от кода, позволяют запускать функции с различными аргументами и в различных условиях, сравнивая ожидаемый и полученный результат. Автоматические тесты делятся на модульные, интеграционные и сквозные.

Модульное тестирование позволяет тестировать отдельные модули кода, независимо друг от друга. Такие тесты выполняются разработчиками параллельно разработке самих модулей. Одновременно с выявлением ошибок, модульное тестирование может выполнять функцию документации, позволяя другим программистам ознакомиться с основными функциями программной единицы. Этот подход является первым уровнем тестирования [1]. Однако в настоящее время набирает популярность другой метод тестирования первого уровня – тестирование, основанное на свойствах, о котором пойдёт речь в статье.

1. Что такое тестирование, основанное на свойствах

Тестирование, основанное на свойствах (Property-based testing, PBT) – это подход к написанию тестов, основанный на автоматизированной проверке инвариантов. В этом случае разработчик описывает свойства своей программы/модуля/функции в терминах некоторых (во многом математических) инвариантов, то есть условий, которые должны оставаться неизменными. Проверку таких инвариантов и поиск кратчайших последовательностей входных данных, нарушающих инвариант, берет на себя соответствующий фреймворк.

Впервые PBT появился в языке Haskell в виде фреймворка QuickCheck.. Несмотря на то, что Property-based testing не получил такую большую известность, как, например, модульное тестирование, в настоящее время фреймворки, похожие на QuickCheck, разработаны для большинства современных языков программирования.

Библиотеки, реализующие PBT, обычно состоят из двух частей: Runners (функции, запускающие тестирование и проверяющие истинность предикатов) и Arbitraries (функции, генерирующие псевдослучайные данные и позволяющие упростить найденный пример, при котором тест завершается с ошибкой).

Псевдослучайно сгенерированные данные – это очень важная часть PBT, которая позволяет воспроизвести фиксированный набор данных для каждой стартовой точки генератора. Такие данные создаются по определяемым шаблонам (разработчик может задавать тип, отношения, граничные значения и многое другое), а для каждого контрпримера (набор данных, который вызывает ошибку в модуле) найти минимальные возможные значения

(например, функция деления может не работать уже на делителе равном 0, а функция сортировки может не учитывать отрицательные числа, то есть самое маленькое возможное целое отрицательное число это -1).

Ещё одна важная особенность тестирования, основанного на свойствах – это абстрактность тестов, такой подход представляет собой более математизированную версию Unit-тестирования. За основу свойств для проверки обычно берутся инварианты реализованного модуля. В связи с тем, что разработчик проверяет только некоторые свойства реализованного модуля, проверить полную работоспособность такого кода очень сложно.

2. Постановка задачи

В качестве примера такого тестирования рассмотрим задачу, часто встречающуюся на собеседованиях для начинающих JavaScript-разработчиков.

В JavaScript существует коллекция Map [1,2], это коллекция ключ/значение, позволяющая использовать ключи любого типа. Необходимо реализовать собственную коллекцию MyMap, позволяющая выполнять запись значения по ключу map.set(key, value), способная возвращать значение по ключу map.get(key) и количество элементов внутри коллекции.

Задача не является сложной в реализации и может быть легко протестирована с помощью ручного или модульного тестирования. Однако можно воспользоваться методом «черного ящика», и с помощью PBT проверить свойства, которые должны выполняться в любых версиях коллекции Map.

Для выполнения данной задачи был использован язык JavaScript, библиотека fast-check в качестве реализации PBT-подхода, и библиотека Jest для запуска тестов.

3. Реализация

3.1. Выделение свойств

Тестирование методом «черного ящика» позволяет не опираться на внутреннюю реализацию компонента и проверять лишь необходимые требования. В приведенных тестах будет использоваться собственная реализация коллекции Map, которая при необходимости может быть заменена на любую другую (включая встроенную в JavaScript).

Для проведения тестирования необходимо выделить какие-нибудь свойства коллекции, которые должны выполняться для любых входных данных. Свойства основаны на требованиях, описанных для реализуемого модуля [4]. Задача выделения свойств является одной из основных в процессе реализации подхода PBT.

Не претендуя на полноту, попробуем ради примера выделить несколько инвариантов относительно реализации Map, чтобы посмотреть, как на их базе сделать тесты.

В первую очередь, коллекция MyMap обязана иметь возможность записывать в неё данные по ключу, и получать данные, соответствующие заданному ключу. То есть, применив к коллекции метод `set(1, true)`, метод `get(1)` обязан возвратить значение `true`. При этом количество элементов внутри коллекции должно соответствовать количеству уникальных ключей.

Количество уникальных ключей можно проверить при помощи встроенной коллекции Set, содержащей в себе только уникальные значения. Свойство количества является инвариантой для коллекции Map, то есть количество ключей в коллекции обязательно меньше или равно количеству добавлений данных в неё (`count(keys) <= count(inserts)`).

Второе важное свойство – это соблюдение уникальности ключей, так как в языке JavaScript присутствуют некоторые особенности, касающиеся представления объектов в

качестве ссылок. Например, объекты `const o1 = {a: 1}` и `const o2 = {a: 1}` при сравнении `a === b` вернут значение `false`, так как `o1` и `o2` являются ссылками, которые указывают на разные объекты. Точно так же сравнение `[] === []` вернет `false` [3].

Третье свойство – при перезаписи значения по одному и тому же ключу, возвращаться должно только последнее значение.

Проанализировав требования к реализации Map, можно выделить вспомогательные инварианты для дополнительных проверок:

- количество ключей в коллекции обязано быть равно количеству значений в ней (`count(keys) == count(values)`);
- количество ключей в коллекции не может быть отрицательным (`count(keys) >= 0`).

Стоит отметить, что в языке JavaScript существует много особенностей при работе с дробными числами, связанных с точностью и округлением (это связано с представлением чисел в числовом формате IEEE-754). Данные особенности влияют и на работу встроенной коллекции Map, поэтому такие случаи не будут считаться ошибкой и не нуждаются в дополнительной проверке.

Также распространенным свойством является подход эталонной реализации, который будет рассмотрен позже [5].

3.2. Вспомогательные Arbitraries

В реализации будут использованы вспомогательные генераторы свойств (Arbitraries):

```
const fc = require("fast-check");

const anythingWithKeys = () =>
  fc.anything({
    key: fc.base64String({ minLength: 1, maxLength: 10 }), maxDepth: 2,
  });

const objectWithKeys = () =>
  fc.object({
    key: fc.base64String({ minLength: 1, maxLength: 10 }), maxDepth: 2,
  });
```

Генератор `anythingWithKeys` позволяет создать данные любого типа, но в случае создания объекта, его ключи будут создаваться, используя только символы base64 (латинские, цифры, + и /) и уровень вложенности 2 (здесь - только объекты со значениями простых типов) [6, 7].

Генератор `objectWithKeys` позволяет создать объекты, их ключи будут создаваться, используя только символы base64 и уровень вложенности 2 (здесь - только объекты со значениями простых типов или значениями аналогичных генератору `anythingWithKeys`) [6,7].

3.3. Проверка записи значений и их количества

```
const findLast = (arr, callback) => {
  let l = arr.length;
  while (l--) {
    if (callback(arr[l])) return arr[l];
  }
  return -1;
};
```

```
test("Map should contain the same items for all unique items", () => {
  fc.assert(fc.property(fc.array(fc.tuple(anythingWithKeys(),
    anythingWithKeys())), (data) => {
      const map = new MyMap();
      for (let tuple of data) {
        map.set(tuple[0], tuple[1]);
      }

      expect(map.size).toEqual(new Set(data.map((arr) => arr[0])).size);
        for (let tuple of data) {
          expect(map.get(tuple[0])).toEqual(
            findLast(data, (item) => item[0] === tuple[0])[1]
);}}));});
```

`fc.assert` – запускает тестирование, проверяет значение всех сгенерированных значений и отвечает за минимизацию тестового примера в случае ошибки [8].

`fc.property` – отвечает за описание свойств и принимает в себя предикат проверки свойства [8].

В данном тесте генерируется массив из кортежей любых ключей и значений, создается новый объект коллекции MyMap, записываются в него все значения по ключам. Проверку количества элементов внутри MyMap можно произвести с помощью встроенной коллекции Set (множество только из уникальных значений), так как количество элементов соответствует количеству уникальных ключей.

В последнем цикле проверяется совпадение значений из изначальных данных и значений в MyMap по одинаковому ключу. Сравнение производится по последнему элементу в массиве, так как при совпадении ключей, значение перезаписывается. Метод `findLast` является написанным собственноручно, однако начиная с версии NodeJS 18.0.0 он является встроенным в объект Array [2].

В случае возникновения ошибки система сообщит минимальный найденный набор значений для воспроизведения, а также предоставит начальный ключ для повторной генерации таких же данных. Этот ключ можно использовать в качестве параметра `seed` в `fc.assert`.

3.4. Проверка записи уникальных значений

```
test("Map should be able to handle unique object references", () => {
  fc.assert(fc.property(fc.array(fc.tuple(fc.clone(objectWithKeys(), 5),
    fc.uniqueArray(anythingWithKeys(), { maxLength: 5, minLength: 5 }))),
           (data) => {
             const map = new MyMap();
             for (let tuple of data) {
               for (let i = 0; i < tuple[0].length; i++) {
                 map.set(tuple[0][i], tuple[1][i]);
               }
             }
             for (let tuple of data) {
               for (let i = 0; i < tuple[0].length; i++) {
                 expect(map.get(tuple[0][i])).toEqual(tuple[1][i]);
    }}}));});
```

В данном тесте генерируется массив из кортежей, где в качестве ключа выступает набор из 5 одинаковых объектов (важно, что они являются одинаковыми по структуре и содержанию, но представлены разными ссылками), а значениями выступают 5 любых возможных объектов. Создается новый объект коллекции MyMap, записываются в него все значения по уникальным

ключам во вложенном цикле.

Значения для каждого объекта должны быть корректно обработаны. Например, если будут сгенерированы входные данные `data = [[{a:1}, {a:1}], [5, false]]`, то для `myMap.get(data[0][0])` должно быть возвращено значение 5, а для `myMap.get(data[0][1])` – значение `false`.

3.5. Проверка записи для одного ключа

```
test("Map should be able to handle same object references", () => {
  fc.assert(fc.property(anythingWithKeys(),
    fc.uniqueArray(anythingWithKeys(), { maxLength: 10, minLength: 1 }),
              (key, values) => {
                const map = new MyMap();
                for (let value of values) {
                    map.set(key, value);
                }

                expect(map.get(key)).toEqual(values[values.length - 1]);
      }));});
```

В данном тесте генерируются две входных переменных: любое значение (включая массивы и объекты) и массив из любых уникальных значений длиной от 1 до 10. Создается новый объект коллекции MyMap, записываются в него все значения первой переменной.

Коллекция MyMap должна по уникальному ключу вернуть последнее записанное значение из массива входных данных. По умолчанию fast-check выполняет 100 таких тестов [4], однако это число может быть настроено.

3.6. Проверка эталонной реализации

Так как JavaScript уже имеет встроенную реализацию коллекции, можно воспользоваться ей для проверки функциональности.

```
test("Map should act the same way as JS Map", () => {
  fc.assert(fc.property(fc.array(fc.tuple(anythingWithKeys(),
    anythingWithKeys())),
         (data) => {
           const map = new MyMap();
           const originMap = new Map();
           for (let tuple of data) {
             map.set(tuple[0], tuple[1]);
             originMap.set(tuple[0], tuple[1]);
           }

           expect(originMap.size).toEqual(map.size);
           for (let tuple of data) {
             expect(map.get(tuple[0])).toEqual(originMap.get(tuple[0]));
      }}));});
```

В данном тесте генерируется массив любых переменных для ключей и значений. Создается новый объект коллекции MyMap и новый объект коллекции Map, в них записываются все значения по уникальным ключам.

Для каждого ключа значение в обеих коллекциях должны совпадать, также в них должно совпадать общее число элементов.

Наиболее часто проверяемыми свойствами могут являться также инвариантность (функция сортировки, примененная к массиву, не должна менять его длину), идемпотентность

(функция сортировки, примененная к отсортированному массиву, не должна ничего менять), инверсии (например, метод дешифрования, примененный после метода шифрования, должен возвращать исходные данные), обратимости (функция переворачивания массива, примененная дважды, должна возвращать исходный массив). Также PBT может использоваться для проверки устойчивости модуля (что бы не было введено в модуль, он не должен завершиться с ошибкой), или для свойств, трудных в реализации, но легко проверяемых (например, сортировки массива).

4. Преимущества и недостатки подхода PBT

Основным преимуществом подхода PBT является компактность тестов. С помощью пары строк кода можно проверить тысячи возможных вариантов значений и их комбинаций, вычислить множество возможных крайних случаев, которые могли быть упущены при других подходах. Также PBT помогает лучше разобраться в особенностях реализованного модуля при анализе и написании свойств.

Однако тестирование, основанное на свойствах, не лишено и недостатков. Такие тесты писать сложнее, чем обычные, потому что необходимо выводить основные свойства метода, что не всегда бывает очевидным сразу. Подход Property-based testing не так широко известен, поэтому первоначальное ознакомление с ним может быть затруднительным из-за недостатка информации и примеров. Также тестирование проводится на уровне абстракций, тогда как обычные Unit-тесты в целом проще воспринимаются и служат еще и документацией к коду. Важной особенностью является тот факт, что может возрасти время на выполнение тестов, так как они запускаются огромное количество раз.

Необходимо учитывать и то, что выявленные свойства не гарантируют правильность реализации, а могут лишь помочь выявить ошибки в пограничных и неочевидных случаях. В целом подход PBT не может заменить другие методы тестирования, однако точно будет отличным дополнением к ним.

Заключение

В данной статье был рассмотрен подход тестирования, основанного на свойствах. Был проведен анализ основных возможностей подхода, реализовано тестирование методом «черного ящика» для различных свойств собственной реализации коллекции MyMap, выявлены плюсы и минусы, проведена оценка. В статье рассмотрен общий подход в реализации PBT, а также приведен пример реализации на языке JavaScript с использованием библиотеки fast-check.

Литература

1. Закас Н. JavaScript. Оптимизация производительности. – Пер. с англ. – СПб.: Символ-Плюс, 2012. – 256 с. – ISBN 978-5-93286-213-1
2. MDN Web Docs – Mozilla. – Режим доступа: https://developer.mozilla.org. – (Дата обращения: 13.04.2023).
3. Современный учебник JavaScript. – Режим доступа: https://learn.javascript.ru. – (Дата обращения: 13.04.2023)
4. Property-based тестирование для JavaScript и UI: необычный подход к автоматизированным тестам. – Режим доступа: https://habr.com/ru/companies/vk/articles/494110. – (Дата обращения: 14.04.2023)
5. Как перестать беспокоиться и начать писать тесты на основе свойств. – Режим доступа: https://habr.com/ru/articles/434008. – (Дата обращения: 14.04.2023)
6. Property-based тестирование с QuickCheck. – Режим доступа: https://habr.com/ru/companies/typeable/articles/570922. – (Дата обращения: 14.04.2023)

7. Choosing properties for property-based testing. – Режим доступа: https://fsharpforfunandprofit.com/posts/property-based-testing-2. – (Дата обращения: 14.04.2023)
8. Arbitraries. – Режим доступа: https://github.com/dubzzz/fast-check/blob/main/packages/fast-check/documentation/Arbitraries.md#combinators. – (Дата обращения: 14.04.2023)

УДК 519.8

СРАВНЕНИЕ РАЗЛИЧНЫХ МОДИФИКАЦИЙ ГЕНЕТИЧЕСКОГО АЛГОРИТМА НА ПРИМЕРЕ ЗАДАЧИ ОПТИМИЗАЦИИ ФУНКЦИИ ДВУХ ПЕРЕМЕННЫХ

Л. Е. Боголепова

Воронежский государственный университет

Введение

Эвристические алгоритмы – методы решения различных задач, которые, в противопоставление точным методам, находят решение путем ограничения простого перебора различными способами. Такие алгоритмы не гарантируют получение точного ответа, но применяются в особо затратных по вычислительным мощностям задачах, решение которых точным методом не представляется возможным. Одним из представителей эвристических алгоритмов является *генетический алгоритм* [1–5], использующий в решении задач закономерности эволюционного развития живого мира.

Важным критерием оценивания работы эвристических алгоритмов является вычисление решения с приемлемой точностью и за приемлемое время. В данной статье проводится вычислительный эксперимент по оптимизации функций двух переменных при помощи двух модификаций генетического алгоритма, в ходе которого проводится сравнение точности и времена выполнения для рассматриваемых модификаций.

1. Описание алгоритма

Генетический алгоритм позволяет получить приближенное решение задачи при помощи следующих операторов: создания *популяции*, подбора родительских *особей*, скрещивания этих особей, *мутации* полученных потомков. Таким образом, популяция становится все более однородной, и в конце концов получаем наилучшую особь, которая достаточно хорошо удовлетворяет поставленной задаче. Здесь *особь* или *генотип* – одно из множества возможных решений, называемого *популяцией*. Также для решения задачи при помощи генетического алгоритма необходима *функция приспособленности,* позволяющая оценить, насколько данное решение приблизилось к точному.

Было принято решение осуществить выбор родительских генотипов методом *пропорциональной селекции*. В этом методе вероятность выбора решения x^i в качестве одного из родителей определяется следующим образом:

$$p(x^i) = \frac{f(x^i)}{\sum\limits_{x^j \in I} f(x^i)},$$

где $f(x^i)$ – приспособленность особи x^i.

В настоящей работе рассматриваются две модификации генетического алгоритма, различающиеся оператором скрещивания. Будем называть их Генетический алгоритм 1 и Генетический алгоритм 2.

Оператор скрещивания для Генетического алгоритма 1 реализует *арифметический кроссинговер*. Строится потомок H родителей $H1$ и $H2, H_1 = \left(h_1^1, h_1^2, ..., h_1^n\right), H_2 = \left(h_2^1, h_2^2, ..., h_2^n\right)$, такой что:

$$h^i = wc_1^i + (1-w)c_2^i,$$

где $w \in [0,1]$ – случайное число.

Оператор скрещивания для Генетического алгоритма 2 реализует *эвристический кроссинговер*. Строится потомок H родителей $H1$ и $H2, H_1 = \left(h_1^1, h_1^2, ..., h_1^n\right), H_2 = \left(h_2^1, h_2^2, ..., h_2^n\right)$, такой что:

$$h^i = w(c_1^i - c_2^i) + c_1^i,$$

где $w \in [0,1]$ – случайное число.

Мутация – замена случайного гена данной особи на случайное число. В настоящей работе вероятность мутации принята равной 0.000001. В новую популяцию попадают лишь дочерние особи, родительские же – утрачиваются.

Алгоритм останавливает свою работу, когда разница между приспособленностью лучшей и худшей особи популяции становится не больше заданного числа ε. Тогда за ответ принимается лучшее решение полученной популяции.

2. Вычислительный эксперимент

В настоящей работе для тестирования генетического алгоритма используются функции, представленные в таблице 1. Данные функции являются классическими тестовыми функциями для методов оптимизации [6].

Таблица 1

Тестовые функции

Название функции	**Формула**
Функция Экли	$f(x,y) = -20e^{(-0.2\sqrt{0.5(x^2+y^2)})} - e^{0.5(\cos(2\pi x)+\cos(2\pi y))} + e + 20$
функция Била	$f(x,y) = (1.50x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$
функция Бута	$f(x,y) = (x + 2y - 7)^2 + (2x + y - 5)^2$
функция Матьяса	$f(x,y) = 0.26(x^2 + y^2) - 0.48xy$
функция Химмильблау	$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)$
функция Гольдштейна-Прайса	$f(x,y) = (1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)) \times$ $\times(30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2))$
функция Леви №13	$f(x,y) = \sin^2 3\pi x + (x-1)^2(1 + \sin^2 3\pi y) +$ $+(y-1)^2(1 + \sin^2 2\pi y)$
функция Трехгорбого верблюда	$f(x,y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2$

функция Шаффера №2	$f(x,y)=0.5+\frac{\sin^2(x^2-y^2)-0.5}{(1+0.001(x^2+y^2))^2}$

Будем проводить эксперимент при различных значениях параметров N и ε, где N – кол-во особей в популяции, а ε – точность, при которой происходит остановка алгоритма. Так как генетические алгоритмы могут отрабатывать с разными точностью и скоростью, алгоритм запускался по 10 раз и вычислялось среднее время поиска минимума функции и средняя точность решения. В таблице 2 приведены результаты выполнения программы с разными входными данными.

Таблица 2

Результаты вычислительного эксперимента при различных значениях параметров алгоритма

№	N	ε	Генетический алгоритм 1		Генетический алгоритм 2	
			Средняя точность решения	Среднее время решения (с)	Средняя точность решения	Среднее время решения (с)
1	1000	0.0001	0.1663482	0.059222	0.1948927	0.027988
2	3500	0.00001	0.0611695	0.822488	0.0515569	0.391977
3	6500	0.000001	0.0474737	2.833766	0.0398587	1.497599
4	10000	0.0000001	0.0165087	7.510944	0.0225008	3.886000
5	16000	0.0000001	0.0110112	10.68263	0.0160327	10.59305
6	18000	0.0000001	0.0066785	18.2888	0.0074152	15.19944

Заметим, что при повышении количества особей в популяции повышается точность вычислений, но также значительно растет время вычислительного эксперимента.

Также заметим, что точность работы одного и того же метода может отличаться от функции к функции. В таблице 3 представлены результаты вычислительного эксперимента для приведенных выше тестовых функций.

Таблица 3

Погрешность оптимизации каждой из рассматриваемых тестовых функций

Функции:	Точность оптимизации (при N = 10000, ε = 0.000001)	
	Генетический алгоритм 1	Генетический алгоритм 2
Экли	0.012585217933352766	0.03972444075639103
Била	0.013524181924560508	0.013514657132823314
Бута	1.919915631442378E-10	3.953874981030927E-10
Гольдштейна-Прайса	1.2460832365945862E-10	7.903455667701564E-12
Химмельблау	1.8799170482012888E-15	1.294454105368583E-11
Леви №13	7.297365445112557E-4	5.0222436905857875E-6
Матьяса	1.3460598370297588E-14	7.526729182454928E-9
Шаффера №2	2.220446049250313E-16	7.214578934267024E-11
Трехгорбого верблюда	1.0603703555830562E-5	5.000651543604403E-9

Визуализируем зависимость точности оптимизации времени выполнения задачи от количества особей в популяции на графике. На рисунке 1 изображена зависимость времени

работы алгоритма от количества особей в популяции. На рисунке 2 – зависимость погрешности решения от количества особей.

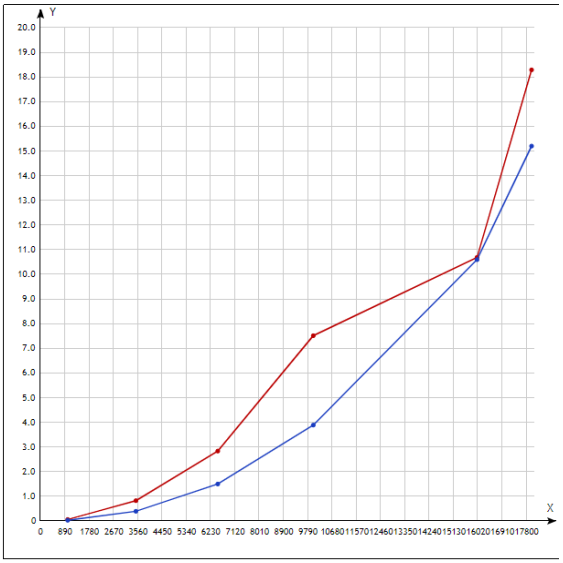


Рис. 1. Зависимость времени выполнения вычислений от N. Красным цветом обозначены результаты для Генетического алгоритма 1, а синим – для Генетического алгоритма 2

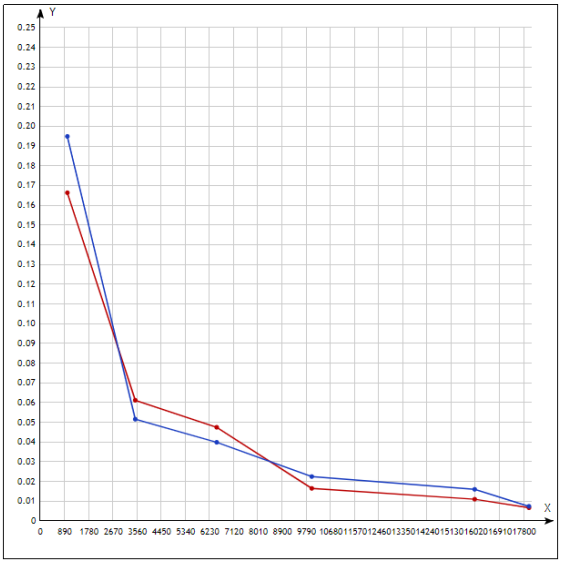


Рис. 2. Зависимость точности решения от N. Красным цветом обозначены результаты для Генетического алгоритма 1, а синим – для Генетического алгоритма 2

Проанализировав данные, полученные в результате эксперимента, можно сделать вывод о том, что оба алгоритма демонстрируют примерно одинаковую точность, при этом Генетический алгоритм 2, использующий эвристический кроссинговер, в большинстве случаев позволяет получить решение за меньшее количество времени, чем Генетический алгоритм 1.

Список литературы

1. Батищев Д.И. Генетические алгоритмы решения экстремальных задач: учебное пособие / Д.И. Батищев. – Воронеж, 1995. – 64 с.
2. Гладков Л.А. Генетические алгоритмы / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик. – Москва: Физматлит, 2006. – 320 с.
3. Курейчик В.В. Теория эволюционных вычислений / В.В. Курейчик, В.М. Курейчик, С.И. Родзин. – Москва: Физматлит, 2012. – 260 с.
4. Курейчик В.М. Эволюционные методы решения оптимизационных задач / В.М. Курейчик. – Таганрог: Издательство ТРТУ, 1999. – 95 с.
5. Генетический алгоритм: Википедия. – URL: https://ru.wikipedia.org/wiki/ Генетический_алгоритм.
6. Тестовые функции для оптимизации: Википедия – URL: https://ru.wikipedia.org/wiki/Тестовые_функции_для_оптимизации.

УДК *004.62*

СРАВНЕНИЕ РЕАЛИЗАЦИЙ ПАТТЕРНА «САГА» В МИКРОСЕРВИСНОЙ АРХИТЕКТУРЕ

Е. А. Бодрова

Воронежский государственный университет

Введение

Актуальным решением для больших и сложных веб-приложений считается микросервисная архитектура [1]. В качестве примера компаний, активно использующих микросервисный подход можно привести Netflix, Amazon, PayPal, eBay. Эта парадигма получила широкое распространение среди приложений таких банков как Росбанк, ВТБ, Альфа-Банк. Микросервисная архитектура — это модель, представляющая собой быстро развивающуюся систему, состоящую из множества небольших, связывающихся между собой компонентов (микросервисов). Микросервис — это небольшой и автономный сервис, смоделированный вокруг бизнес-области и, как правило обладает собственной базой данных. Основным преимуществом данной архитектуры является слабая связанность сервисов, благодаря которой возможны независимые развертывание и поддерживаемость сервисов[1, 2], а также простое масштабирование всей системы. Однако для бизнес-процессов, включающих несколько микросервисов, требуется координация сервисов, необходимая для обеспечения согласованности и выполнения определенных транзакционных гарантий[2].

Паттерн Сага (*Saga или повествование*) часто упоминается в качестве решения проблемы организации согласованности данных в микросервисной архитектуре. Такой подход разделяет транзакцию на несколько локальных транзакций, так что блокировки для включенных ресурсов не нужно удерживать до полного завершения. Тем не менее, данный паттерн имеет сложность, поскольку такие независимые локальные транзакции должны быть скоординированы, а компенсация должна быть возможна, чтобы учесть различные сценарии отказа. В данной работе исследуются различные существующие подходы к реализации путем сравнения их возможностей на основе определенного списка критериев.

1. Паттерн Сага

Паттерн Сага представляет собой надстройку над другим подходом разработки микросервисной архитектуры — паттерном «одна база данных на сервис». В таком подходе база данных сервиса фактически является частью его реализации. Другие сервисы не могут получить к базе данных данного сервиса прямой доступ. Пример организации представлен на рис.1.

Рис. 1. Подход «база данных на сервис»

Однако обеспечение слабой связанности таким подходом создает проблему для реализации бизнес-транзакций, объединяющих несколько сервисов и запросов для данных из нескольких баз данных. Поэтому необходим механизм для реализации транзакций, которые охватывают сервисы. Данный подход использует вместо ACID-транзакций операцию, охватывающую несколько сервисов и стремящуюся поддерживать согласованность данных и использующая то, что называется повествованием (или сагой), — последовательность локальных транзакций на основе сообщений [1, 2]. Одна из проблем повествований связана с тем, что по своей природе они являются ACD (Atomicity, Consistency, Durability — «атомарность, согласованность, долговечность»). Им не хватает поддержки изолированности, которая есть в ACID-транзакциях. В итоге приложение должно использовать методики проектирования, которые устраняют или снижают влияние аномалий конкурентности, вызванных нехваткой изолированности. Самыми распространенными методиками являются методы «хореография» и «оркестратор».

1.1. Оркестрация

Оркестрация — это еще один способ реализации повествований [1]. Она подразумевает определение класса-оркестратора. Оркестратор определяет дальнейшие шаги всего повествования, рассылая инструкции микросервисам, участвующим в бизнес-процессе. Пример работы оркестратора можно увидеть на рис. 2. Оркестратор посылает микросервису B команду (1) и получает ответы от микросервисов (2). Затем отправляет команду микросервису C (3) и D (4).

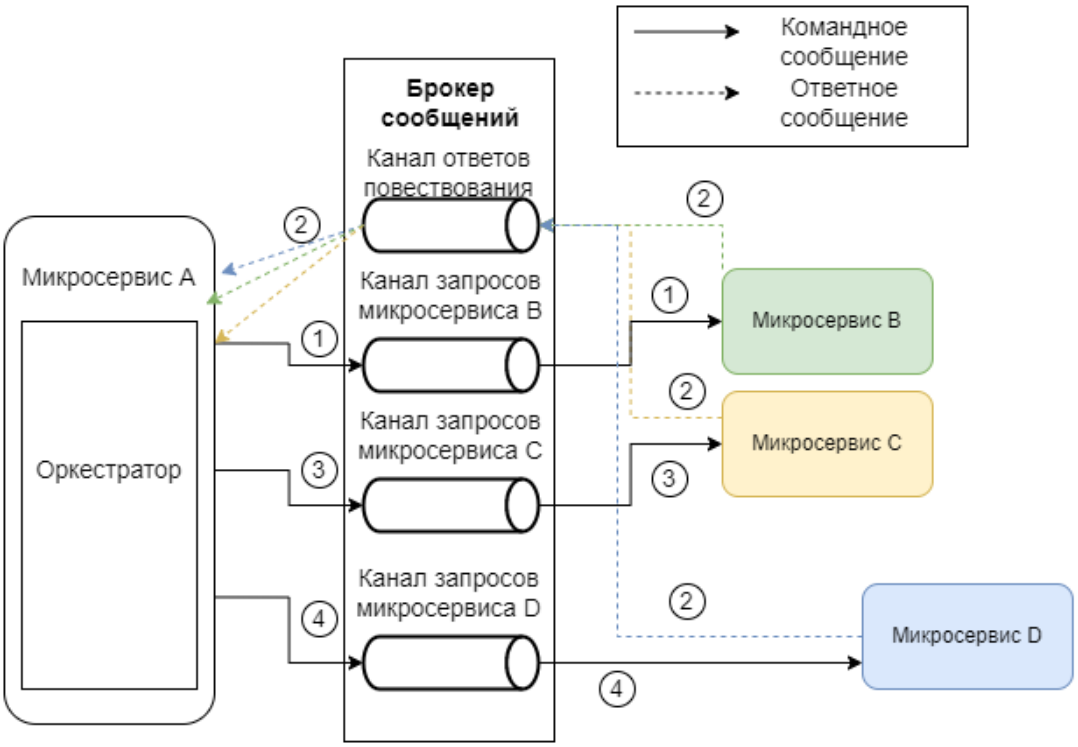


Рис. 2. Реализация повествования при помощи оркестратора

1.1. Хореография

Хореография — это один из способов реализации повествований [2]. Она не предусматривает центрального координатора, который выдает участникам команды. Вместо этого участники подписываются на события друг друга и реагируют соответствующим образом. Пример такого взаимодействия представлен на рис. 3, где под микросервисом понимается связка «микросервис — база данных».

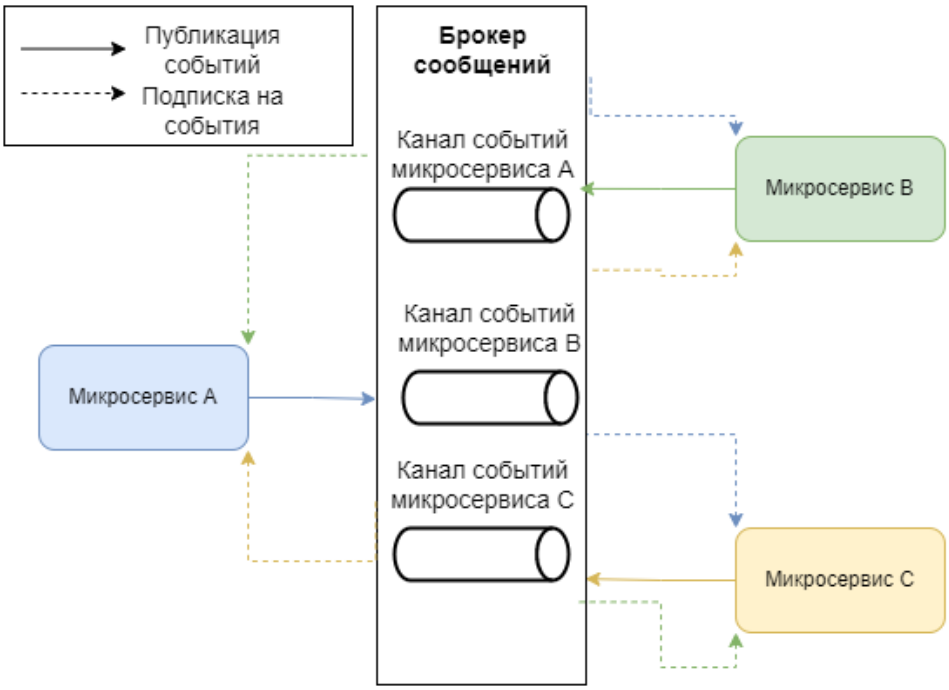


Рис. 3. Реализация повествования при помощи хореографии

2. Сравнение подходов к реализации паттерна Сага

Сравнивая подходы, важно отметить, что оркестрация легче в освоении и приложении, однако слабо подходит для описания сложных взаимодействий независимых систем или для описания процессов высокого уровня абстракции [3]. Хореография, напротив, отстраняясь от описания самого процесса, вводит правила и условия взаимодействия его участников. При этом условия взаимодействия могут быть сужены до описания потоков информации между участниками. Именно поведение участников и потоки информации между ними определяют роль глобального процесса.

В табл. 1 приведено сравнение хореографии и оркестрации с использованием критериев, основанных на архитектурных особенностях каждого. Как видно из табл.1, оркестрация более выгодна в ситуациях с существующим кодом, например, в проектах, находящихся в переходной стадии от монолитной архитектуры к микросервисной [4]. Она использует более простые механизмы реализации большинства критерий благодаря собственной централизации. В отличие от оркестрации, хореография представляет собой более гибкое решение с точки зрения управления микросервисов, и эта гибкость становится усложняющим фактором, например, в вопросе настройка мониторинга или управления ошибок системы.

Таблица 1

Сравнение подходов на основе архитектурных критериев

Критерий	Оркестрация	Хореография
Масштабируемость	+	–
Гибкость	–	+
Мониторинг	Простой механизм	Может требовать более сложного мониторинга, так как каждый микросервис может принимать решения о своих действиях
Управление ошибками	Простой механизм	Может требовать более сложного управления ошибками
Сложность реализации	Хорошо подходит для проектов с легаси-кодом; простота реализации из-за наличия оркестратора	Хорошо подходит для новых проектов; реализация компенсирующих транзакций может быть сложна при большом количестве микросервисов

Кроме того, можно определить следующие технические критерии для сравнения:

1. Время выполнения распределенной транзакции. Время обработки оркестрацией значительно выше, чем хореографией при условии использования транзакции с одним и тем же набором участвующих микросервисов [5]. Это связано с тем, что в оркестрации должны проверяться все ресурсы и данные одновременно, тогда как в хореографии выполнение транзакции происходит последовательно, от одного микросервиса к другому, что ускоряет общий процесс, так как нет необходимости обновлять все ресурсы сразу.
2. Коммуникационные затраты. В независимости от подхода, микросервисам необходимо обмениваться сообщениями друг с другом. Очевидно, что такие затраты будут выше у оркестрации, так помимо взаимодействия микросервисов друг с другом и инициирующих сообщений присутствует коммуникация с оркестратором, что увеличивает коммуникационные расходы.
3. Компенсационные транзакции. Время выполнения таких транзакций приблизительно одинаковы как для оркестрации, так и для хореографии [5].

Заключение

Таким образом, были определены критерии сравнения подходов к организации паттерна «Сага». Основываясь на приведенном анализе можно сделать вывод о том, что оркестрация больше подходит для проектов с уже существующей кодовой базой, так как этот подход прост в реализации и легко масштабируем, тогда как хореография представляет собой более гибкий подход к управлению микросервисами. Преимущества хореографии становятся заметны при рассмотрении технических критериев сравнения. Так, хореография имеет наименьшее время выполнения распределенной транзакции за счет отсутствия дополнительных расходов на взаимодействие с центральным элементом, как в случае с оркестрацией. По аналогичной

причине хореография выполняет транзакцию за наименьшее количество сообщений. Подобные преимущества получены данным подходом за счет наибольшего уровня абстракции, который используется при реализации, в отличие от оркестрации.

Каплиева Наталья Алексеевна (научный руководитель): канд. физ.-мат. наук, доц., доцент кафедры математического обеспечения ЭВМ Воронежского государственного университета.

Литература

1. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга / К. Ричардсон. – Санкт-Петербург : Питер, 2019. – 544 с.
2. Microsevices. – Режим доступа: https://microservices-io (Дата обращения: 01.03.2023).
3. Артамонов И. В. Оркестровка и хореография / И. В. Артамонов // Моделирование бизнес-транзакций. – Иркутск : Издательство БГУ, 2016. – С. 39–45.
4. Dürr K., Lichtenthaeler R., Wirtz G. Saga Pattern Technologies: A Criteria-based Evaluation // 12th International Conference on Cloud Computing and Services Science. – ISBN 978-989-758-570-8; ISSN 2184-5042, 2022. – С. 141–148.
5. Maamar Z., Asim M., Cheikhrouhou S., Qamar A. Orchestration- and choreography-based composition of Internet of Transactional Things // Service Oriented Computing and Applications. – 2021. – С. 157–170.

УДК 004.4

СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЯ ПО УЧЕТУ КРУПНОГО РОГАТОГО СКОТА

В. А. Болдорев

Воронежский государственный университет

Введение

Целью выполняемой работы является создание web-приложения, которое нацелено на облегчение, цифровизацию и автоматизацию основных дисциплин, выполняемых малыми фермерскими хозяйствами, которые на данный момент выполняются на бумаге.

1. Технические требования

Приложение включает следующие модули:
1. «Моя ферма».
2. «Учет».
3. «Планирование».
4. «Управление».
5. «Пользователь».

Модуль «Моя ферма» обеспечивает следующими возможностями:
1. Возможность просмотра статистических данных.
2. Возможность редактирования фермы.

Модуль «Учет» обеспечивает возможностью производить учет по следующим темам:
1. Учет надоев.
2. Ветеринарный учет.
3. Количественный учет.
4. Весовой учет.
5. Учет воспроизводства.

Модуль «Планирование» обеспечивает следующими возможностями:
1. Возможность просмотра запланированных событий в календаре.
2. Возможность планировать новые события.

Модуль «Управление» обеспечивает следующими возможностями:
1. Возможность добавлять новых животных.
2. Возможность редактировать уже существующих животных.
3. Возможность просмотра генеалогического древа по каждому животному.
4. Возможность импортировать/экспортировать животных.

Модуль «Пользователь» обеспечивает следующими возможностями:
1. Возможность просмотра информации о пользователе.
2. Возможность редактирования основной информации.
3. Возможность выйти из системы.

Осуществляется ограничение функциональности в зависимости от роли пользователя.

2. Анализ функциональности

На этапе проектирования было выделено 3 основных роли: контролёр надоев, контролёр состояния животных, администратор. В результате анализа функциональности приложения, были определены сценарии взаимодействия пользователей с программным продуктом, которые отображены в Use Case диаграммах.

Контролёр надоев может просматривать основную статистику фермы без возможности редактирования, производить работу с календарем и осуществлять учет надоев (рис. 1).

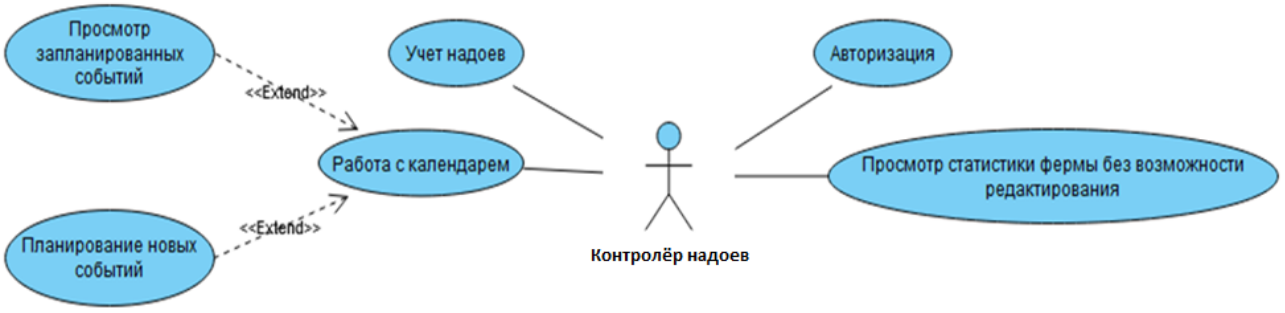


Рис. 1. Диаграмма вариантов использования для контролёра надоев

Контролёр состояния животных может просматривать основную статистику фермы без возможности редактирования, производить работу с календарем, производить ветеринарный учет, количественный учет, весовой учет и учет воспроизводства (рис. 2).

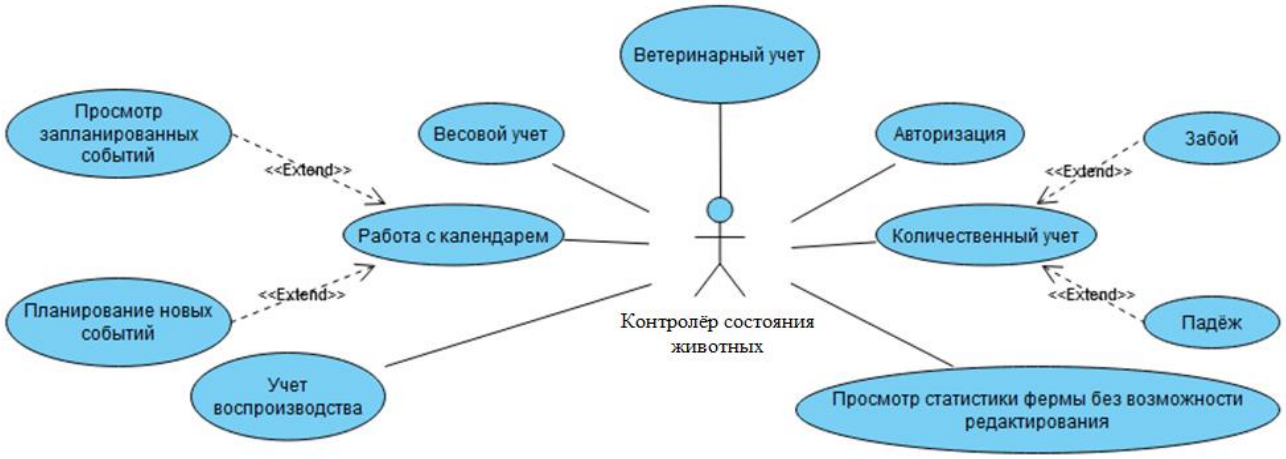


Рис. 2. Диаграмма вариантов использования для контролёра состояния животных

Администратор может выполнять действия над пользователями, просматривать статистику фермы, производить работу с календарем, производить всевозможные учеты и редактировать ферму (рис. 3).

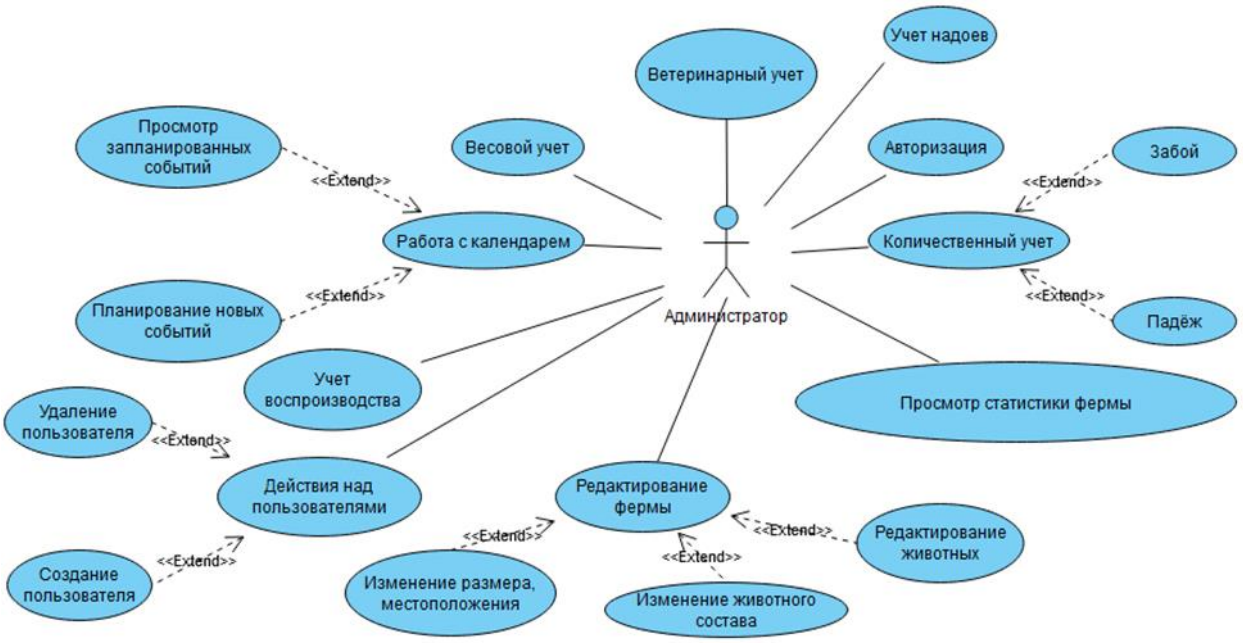


Рис. 3. Диаграмма вариантов использования для администратора

3. Реализация

Web-приложение разрабатывалось в среде Visual Studio 2022. В качестве языков программирования были выбраны C# и JavaScript. Использовались фреймворки ASP.NET Core и jQuery. ASP.NET Core был выбран благодаря его кроссплатформенности, что позволяет разворачивать приложение на максимально возможном спектре устройств и операционных систем, а не только на Windows. Для хранения данных была выбрана свободная объектно-реляционная система управления базами данных – PostgreSQL. Объектами, хранимыми в базе данных, являются сущности «Death», «Slaughter», «User», «Animal», «Address», «Farm», «Reproduction», «Vaccinations», «WeightAccounting», «MilkAccounting», «Event» (рис. 4).

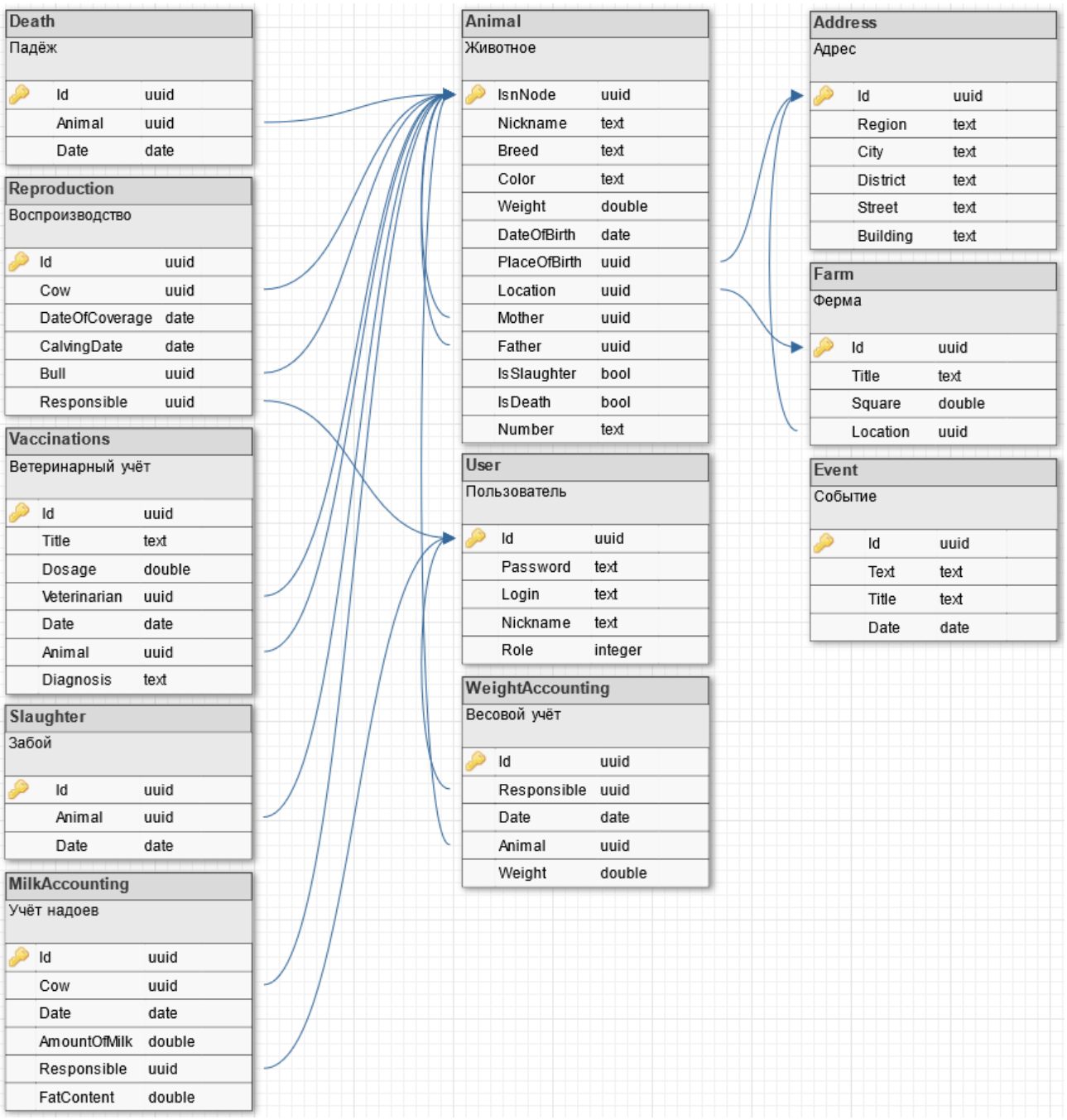


Рис. 4. Схема базы данных

Для работы с календарем, в web-приложение была внедрена JavaScript-библиотека FullCalendar.

Заключение

В результате проделанной работы было создано web-приложение, которое отвечает всем требованиям постановки задачи.

Полученное web-приложение рассчитано на малые фермерские хозяйства и позволяет перенести часть дисциплин, таких как основные типы учетов, планирование событий и сбор статистики по ферме, которые проводятся в бумажном виде, в цифровой формат и автоматизировать их.

Литература

1. Албахари Джозеф, Албахари Бен. C# 7.0 Справочник. Полное описание языка.: Пер. с анг. – СПб. : ООО «Диалектика», 2019. – 1024 с.

2. Документация по ASP.NET Core. – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-7.0 (Дата обращения: 20.03.2023).

3. Документация по FullCalendar. – Режим доступа: https://fullcalendar.io/docs (Дата обращения: 10.04.2023).

УДК 004.94

ИСПОЛЬЗОВАНИЕ BSP-ДЕРЕВЬЕВ ДЛЯ ГЕНЕРАЦИИ КАРТЫ ПРИ РАЗРАБОТКЕ ИГРЫ-БРОДИЛКИ, ВКЛЮЧАЮЩЕЙ ВЫБОР ПРАВИЛЬНОГО ПЕРЕВОДА АНГЛИЙСКИХ СЛОВ

Д.О. Бутузова, Е.В. Трофименко

Воронежский государственный университет

Введение

Современные технологии предоставляют множество возможностей для создания обучающих игр, которые помогают улучшить знания и навыки игроков в различных областях. В данной статье мы представляем идею обучающей игры-бродилки с использованием BSP-деревьев для генерации карты и добавлением уровней с выбором правильного перевода английских слов. Эта игра предназначена для детей младшего школьного возраста, которые хотят улучшить свои знания английского языка и провести время с удовольствием.

Создание обучающих игр может быть эффективным методом обучения для детей. Это связано с тем, что такой подход позволяет детям учиться через интерактивный опыт и игру, что может значительно улучшить усвоение знаний. Наша обучающая игра, созданная на базе движка Unity 3D, помогает игрокам изучать английский язык и развивать свои когнитивные способности.

1. Среда разработки.

Unity 3D [1] - это кроссплатформенный движок для разработки игр, который позволяет создавать игры для разных платформ, таких как Windows, macOS, Linux, iOS, Android и многих других. Unity 3D используется для создания игр различных жанров, включая экшн, приключения, стратегии, головоломки и многие другие.

Одной из главных особенностей Unity 3D является его простота использования и широкие возможности для настройки и настройки игровых объектов и сцен. Для разработки игр в Unity 3D используется язык программирования C#, который является одним из самых популярных языков программирования в мире.

Благодаря своей популярности и широким возможностям, Unity 3D используется как независимыми разработчиками, так и крупными игровыми студиями для создания игр различных жанров и сложности.

2. Генерация карты.

Одной из основных особенностей игры является генерация карты на основе BSP-деревьев. BSP-деревья - это деревья, которые используются для разбиения двумерной плоскости на области [2, 3]. В нашей игре мы используем BSP-деревья для генерации уровней, что позволяет создавать большое количество различных уровней. Это позволяет создавать уникальный игровой опыт для игроков и повышать интерес к игре.

Бинарное пространственное разбиение (BSP) - это техника разбиения пространства на две части (обычно левую и правую), используя плоскость, которая проходит через центр масс объекта. BSP-деревья используются в различных приложениях компьютерной графики, в том числе для генерации уровней.

В Unity 3D также можно использовать BSP-деревья для генерации уровней, но они не являются стандартным инструментом для этого. В Unity 3D более распространенным подходом является использование пространственной сетки (Grid) или создание уровней вручную в редакторе сцен (Scene Editor).

Однако, использование BSP-деревья в Unity 3D возможно. Для этого можно воспользоваться сторонними инструментами или библиотеками, которые позволяют создавать BSP-деревья. Но нашей целью является реализация алгоритма, поэтому прибегнем к знаниям программирования и математики.

Алгоритм реализации BSP-дерева следующий:

1. Создайте структуру данных, представляющую пространство уровня, в котором вы хотите создать BSP-дерево. Это может быть двумерный или трехмерный массив, содержащий информацию о каждом блоке, который может быть помещен на уровень.
2. Создайте корневой узел BSP-дерева, представляющий всю область уровня.
3. Разбейте область, представленную корневым узлом, на две части путем выбора случайной плоскости. Эта плоскость будет служить разделителем между двумя частями уровня.
4. Создайте два дочерних узла BSP-дерева, представляющих каждую из двух частей уровня, разделенных плоскостью.
5. Рекурсивно повторите шаги 3-4 для каждого дочернего узла BSP-дерева до тех пор, пока каждый узел BSP-дерева не будет содержать небольшую область уровня.
6. Пройдите через BSP-дерево в порядке обхода в глубину (DFS), и для каждого узла BSP-дерева, отрисуйте блоки уровня в этом узле.
7. При отрисовке блоков уровня, отрисуйте сначала блоки в левом дочернем узле BSP-дерева, затем разделительную плоскость, и затем блоки в правом дочернем узле BSP-дерева. Это поможет избежать отрисовки блоков, которые могут быть закрыты другими блоками.
8. Повторите шаг 6-7 для каждого узла BSP-дерева.

Это лишь общий подход к созданию BSP-дерева, и конкретные шаги могут отличаться в зависимости от вашей конкретной задачи.

Проиллюстрировать алгоритм можно с помощью рис. 1[4]. На нем для упрощения восприятия рассмотрена двумерная квадратная плоскость A.

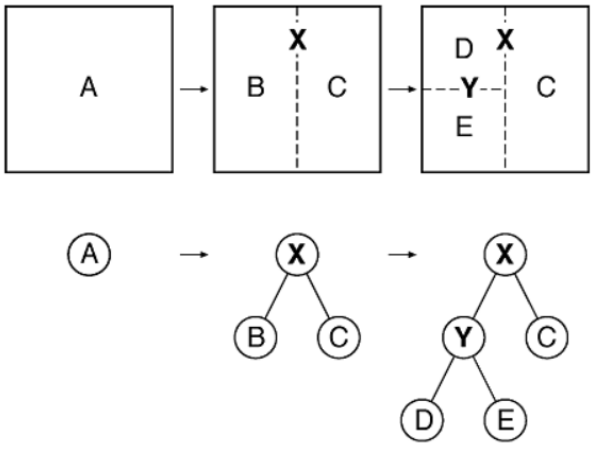


Рис. 1. Результат построения bsp-дерева

Пример области, разделённой на листья приведен на рис. 2:

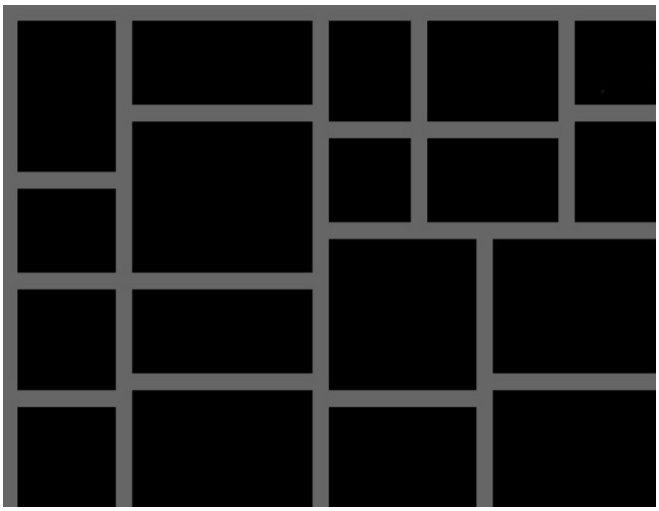


Рис. 2. Пример с листьями, разделёнными линиями

3. Обработка столкновений.

Обработка столкновений в 3D среде Unity является важным аспектом при разработке игр. Она основана на системе физики, которая использует движок PhysX от NVIDIA.

Для обработки столкновений в Unity необходимо добавить компонент коллайдера на объекты, которые должны взаимодействовать друг с другом [5, 6]. Коллайдеры являются геометрическими фигурами, которые определяют границы объекта. Unity поддерживает различные типы коллайдеров, такие как кубы, сферы, капсулы, меш-коллайдеры и другие.

Кроме того, на объекты, которые должны иметь физические свойства, необходимо добавить компонент физического тела. Физическое тело определяет массу, скорость, силу тяжести и другие физические свойства объекта.

Параметры коллайдеров и физических тел настраиваются с помощью Unity Editor. Можно настроить коэффициент трения, который определяет, насколько легко объекты будут скользить друг по другу, а также настроить силу удара, которую они могут наносить друг другу. Эти параметры могут быть использованы для достижения нужного поведения при столкновениях.

При обработке столкновений в Unity можно использовать скрипты, которые реагируют на столкновения между объектами. С помощью скриптов можно определять, какие действия должны происходить при столкновениях, например, отнимать здоровье у персонажа при ударе, перемещать объекты в другую точку, или триггерить анимации.

Также в Unity есть возможность использовать триггеры, которые не взаимодействуют физически, но могут быть использованы для определения того, когда объекты входят или выходят из зоны действия. Для этого на объекты добавляется компонент триггера вместо коллайдера.

В целом, система обработки столкновений в Unity является мощным инструментом для создания интерактивных 3D-сцен, и может быть использована для реализации множества различных задач в различных областях, включая игры, виртуальную реальность и архитектурную визуализацию.

4. Выбор перевода английских слов.

Другой важной особенностью игры является выбор правильного перевода английских слов. На экране появляется английское слово, когда игрок пытается собрать монетку, и игрок должен выбрать правильный перевод из нескольких вариантов ответа.

Слова для каждого уровня будут храниться в базе данных, куда они будут загружены с сайтов с дополнительными материалами к учебникам по английскому языку. Так мы сможем подготовить словарь, соответствующий школьной программе и уровню ребенка.

Установка уровня сложности в обучающей игре очень важна, так как она может повлиять на ее восприятие. Если игра будет слишком сложной, школьники могут быстро потерять интерес, а если будет слишком просто, то процесс не будет достаточно стимулировать игроков к обучению.

В нашей игре мы используем три уровня сложности, которые отвечают возрастным группам нашей целевой аудитории: для 2-го, 3-го и 4-го класса. Каждый уровень сложности содержит набор слов, которые соответствуют программе по английскому языку для соответствующего класса. Например, первый уровень сложности будет содержать слова, которые обычно изучаются во 2-м классе, такие как животные, цвета, формы и т.д. Второй уровень будет с более сложными словами, которые изучаются в 3-м классе, такими как погода, профессии, спорт и т.д., соответственно последний уровень сложности будет еще труднее, с темами, которые изучаются в 4-м классе, такие как история, география и т.д.

Мы предоставляем игрокам возможность выбрать уровень сложности на стартовом экране игры [7]. Это делается с помощью кнопок в меню. После того, как ребенок выбирает уровень сложности, все слова и переводы, которые появляются в течение игры, будут соответствовать этому уровню.

Таким образом, установка уровня сложности игры является очень важным аспектом обучающей игры, который позволяет пользователям настроить игру под свои индивидуальные потребности и уровень знаний английского языка.

Заключение.

В данной статье описали идею создания обучающей игры-бродилки с использованием BSP-деревьев для генерации карты и добавлением уровней с выбором правильного перевода английских слов. Также были рассмотрены основные компоненты игры, такие как генерация карты, управление персонажем, обработка столкновений, добавление уровней с выбором правильного перевода английских слов и установку уровня сложности игры. Обучающие игры могут стать важным инструментом обучения и развития для младших школьников, а также способом облегчить процесс изучения английского языка. Поэтому, разработка таких игр может стать одним из приоритетных направлений в образовании, а использование технологий и инноваций позволит сделать процесс обучения более увлекательным и интерактивным. Данная игра предназначена для того, чтобы помочь школьникам улучшить свои знания английского языка, проводя время с удовольствием.

Литература

1. Unity, платформа разработки в реальном времени | 3D-, 2D- VR- и AR визуализации. [Электронный ресурс]. – Режим доступа: https://unity.com/ru (Дата обращения: 23.03.2023)
2. Документация Unity [Электронный ресурс]. – Режим доступа: https://docs.unity3d.com/Manual/index.html (Дата обращения: 03.04.2023)

3. Хокинг Д. Unity в действии. Мультиплатформенная разработка на C# / Хокинг Д. - Санкт-Петербург: Питер, 2019. – 336 с.
4. BSP-дерево и способы его применения в трехмерной графике [Электронный ресурс]. – Режим доступа: http://eways.narod.ru/HomePage/3d/bsp_tree.htm (Дата обращения: 07.04.2023)
5. Курс John Lemon's Haunted Jaunt: 3D Beginner [Электронный ресурс]. – Режим доступа: https://learn.unity.com/project/john-lemon-s-haunted-jaunt-3d-beginner (Дата обращения: 03.04.2023)
6. Создаём простой зомби-шутер на Unity [Электронный ресурс]. – Режим доступа: https://tproger.ru/articles/sozdajom-prostoj-zombi-shuter-na-unity/ (Дата обращения: 05.04.2023)
7. Бонд Д. Г. Unity и C#. Геймдев от идеи до реализации / Бонд Д. Г., Лемарчанд Р. - Санкт-Петербург: Питер, 2022. – 928 с.

УДК 004.932

МОДЕЛИ РАСПОЗНАВАНИЯ СОСТОЯНИЯ УСТАЛОСТИ ВОДИТЕЛЯ АВТОТРАНСПОРТА

К. Р. Ватутин, О. Ю. Лавлинская

Воронежский государственный университет

Введение

Высокий уровень стресса, сонливость и отсутствие концентрации внимания – основные факторы, определяющие ухудшение состояние здоровья водителя, что может привести к проблемным ситуациям на дорогах и даже к авариям. Одной из задач, которая привлекает внимание в области исследований предотвращения дорожно-транспортных происшествий, является создание устройств и механизмов, способных осуществлять мониторинг и давать оценку поведению водителя.

При вождении соответствие поведения водителя правилам безопасности напрямую связано с личной безопасностью и безопасностью движения на дороге, поэтому мониторинг поведения водителя за рулем является важной задачей. Решение задачи обеспечения безопасного вождения решается разными способами. Во-первых, производители автомобилей предусматривают различные варианты привлечения внимания водителей: дорожная инфраструктура имеет специальную разметку и покрытие, которые предназначены для того, чтобы водитель взбодрился в процессе езды, голосовые помощники, встроенные в навигаторы, бортовые компьютеры оповещают водителя о необходимости отдыха. Во-вторых, современные информационные технологии распознавания и фиксации позволяют распознавать усталость водителя по биометрическим показателям или по ключевым точкам лица. На основании полученной информации о состоянии водителя вырабатывается управляющий сигнал или звуковое оповещение для предотвращения опасной или аварийной ситуации на дороге[1].

Актуальной задачей является разработка инструментальных подходов к оценке состояния водителя на основе моделей идентификации лица с учетом положения глаз, рта, положения головы (поворота, наклона). Рассмотрим существующие модели оценки состояния водителя с целью выбора наилучших моделей оценки усталости водителя автотранспорта в условиях городского движения для реализации собственного программного решения для оповещения водителя о необходимости остановки движения в состоянии сонливости.

1. Оценка качества идентификации лица в моделях распознавания

1.1 Оценка качества входных данных модели

В качестве входных данных алгоритма распознавания лиц качество изображения лица сильно влияет на окончательную точность распознавания. Специально для оборудования для получения изображений, развернутого в кабине водителя, качество изображений, регистрируемых камерой наблюдения неравномерно, - есть качественные и низкокачественные изображения. Низкокачественные изображения лица обусловлены такими факторами, как яркость, контрастность, разрешение, размытие изображения, и шум – это факторы, которые влияют на окончательный эффект распознавания лиц. При чтении кадра видео в устройстве для получения изображения необходимо выполнить алгоритм обнаружения лица и определение ключевых точек лица, чтобы узнать, где находится лицо на изображении, а также ключевые

точки черт лица и контуров лица. Качество лица необходимо оценить и некачественные изображения отфильтровать. Извлечение изображение лиц высокого качества необходимо для всего процесса распознавания.

Рассмотрим технологию оценки качества изображения лиц по трем параметрам: нечеткое лицо, темное лицо и лицо с односторонним освещением. Принцип работы традиционного метода оценки качества не эталонных изображений заключается в непосредственном вводе всего изображения лица для расчета качества изображения после обнаружения лица. Такие методы не могут точно представить качество лица, поскольку качество ключевых областей лица распознавание различается. Изображение лица, извлеченное алгоритмом обнаружения лица, имеет прямоугольную форму, а внешняя часть щек, волосы и другие части, отличные от черт лица в кадре изображения, будут влиять на качество изображения. Поэтому в процессе обработки используется область пикселей, извлеченная алгоритмом определения ключевых точек, в целом, а для получения более точных локальных особенностей, устраняется интерференция неключевых областей в прямоугольной рамке лица и определяется качество изображения в соответствии с чертами лица. В частности, алгоритм не просто использует пиксели в прямоугольной рамке лица в качестве эталона оценки качества изображения, но идентифицирует ключевые точки лица, обрабатывает пиксели изображения в области ключевых точек и извлекает больше локальной информация о чертах лица. Выполнение различных этапов обработки для оценки размытых лиц, темных лиц или лиц с односторонней освещенностью в ключевых областях лица может более эффективно отфильтровывать лица низкого качества в сложных реальных условиях. Этот метод не требует слишком большого количества аппаратных ресурсов, а скорость обработки очень высока [2,3].

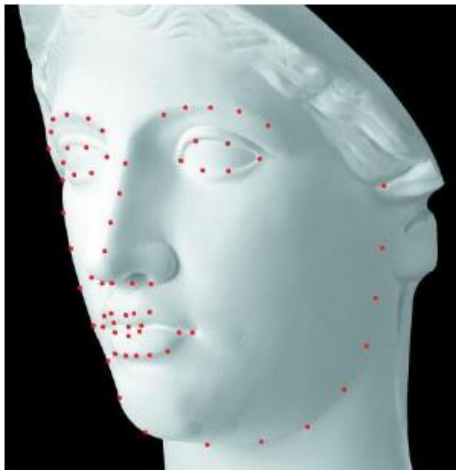


Рис. 1. Схематическая диаграмма ключевых точек лица

На основе пикселей в ключевых областях лица рассчитываются различные показатели, по которым можно определить, является ли лицо размытым, темным, с односторонней освещенностью, и если один из показателей не удовлетворяет критерию, то он оценивается как низкий, тогда и качество изображения также будет низким. Алгоритм определения ключевых точек лица по сравнению с другими моделями распознавания, таким как сравнение с эталонным изображением работает быстрее, занимает меньше ресурсов и устраняет интерференцию областей, не являющихся лицами, и неключевых областей лица. На рис. 1 представлена схема ключевых точек, необходимых для распознавания.

Процесс обучения алгоритму по методу распознавания ключевых точек выглядит следующим образом [4]:

1) Определить количество ключевых точек, например 98 ключевых точек, как в примере на рисунке 1, выбрать наборы изображений лиц разных водителей, применить инструмент маркировки, чтобы вручную отметить 98 точек лица для каждого изображения.

2) Выбрать подходящий алгоритм CNN, обучить на изображении профиля, заданного в качестве входных данных модели алгоритма, получить выходной результат модели, сравнить его с результатом ручной калибровки, рассчитать градиент для обратного распространения ошибки, отрегулировать вес связей, моделировать и непрерывно повторять, обучать N эпох, пока модель не сойдется.

3) Выбрать модель с наилучшей производительностью за N эпох в качестве окончательной модели вывода, подать на вход набор изображений каждого лица, чтобы получить результаты прогнозирования по 98 ключевым точкам.

1.2 Математическая модель обнаружения ключевых точек лица

Рассмотрим алгоритм PFLD, который представляет собой упрощенный алгоритм обнаружения с высокой точностью, небольшой по размерности моделью и высокой скоростью обработки данных, и обеспечивает производительность сверхреального времени на мобильном терминале [3].

PFLD — это практичный детектор ключевых точек лица с высокой точностью в сложных ситуациях, таких как освещение, окклюзия, выражения и неограниченные позы. Алгоритм использует ветвь сети для оценки геометрической интерпретации каждого образца лица, а затем корректирует расположение ключевых точек. Ниже приводится подробное описание функции потерь алгоритма, магистральной сети и вспомогательной подсети.

С точки зрения функции потерь обычно используются следующие варианты: MSE, Softmax + потеря перекрестной энтропии, Smooth L1 [3] и т. д. Чтобы сбалансировать обучающие данные в различных ситуациях и учесть экстремальные ситуации, которые могут возникнуть в реальности (например, обнаружение лиц в случай окклюзии лица) алгоритм может добавлять закрытые лица в данные обучения и обеспечивать согласованность пропорций данных обучения в различных ситуациях. Этот метод может эффективно повысить производительность, контролируя форму выборки данных. И PFLD использует новую функцию потерь для решения проблемы дисбаланса выборки в различных ситуациях. Математическая модель представлена формулой 1:

$$L = \frac{1}{M} \sum_{m=1}^{M} \sum_{n=1}^{N} \left(\sum_{c=1}^{C} w_n^c \sum_{k=1}^{K} \left(1 - \cos \theta_n^k\right) \right) d_{n2}^{m2} \qquad (1)$$

где W_n^c — регулируемая весовая функция (разные веса выбираются для разных ситуаций, таких как нормальная ситуация, ситуация с окклюзией, ситуация с темным освещением и т. д.), θ — трехмерный угол Эйлера позы лица (K =3), d — регрессионный ориентир и метрика Ground-True (как правило, также может быть выбрана метрика MSE, L1) [4].

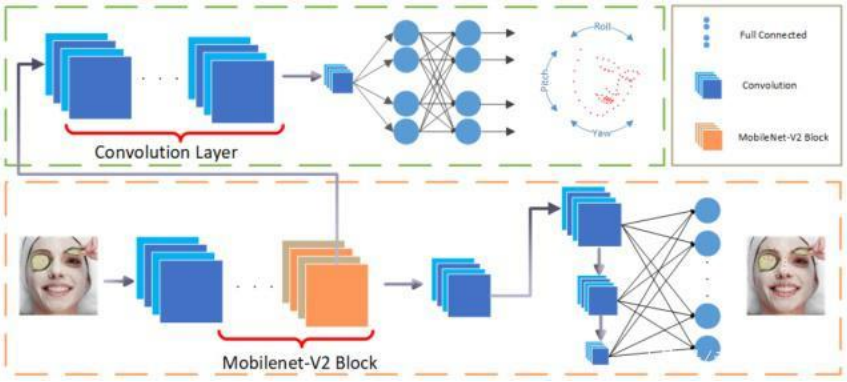


Рис. 2. Общая структура PFLD

Цель построения функции потерь состоит в том, чтобы придать небольшой вес данным с относительно большим размером выборки (таким как фронтальные грани, то есть случай, когда углы Эйлера относительно малы), когда градиент распространяется обратно. Архитектура функции потерь модели решает проблему несбалансированных обучающих выборок, рис. 2. С точки зрения магистральной сети, чтобы удовлетворить потребности устройств с низкой вычислительной мощностью, таких как встроенные или мобильные терминалы, при этом структура сети MobileNet v2 [4] урезана, чтобы уменьшить количество

избыточных параметров, и для извлечения признаков используется упрощенная модель, выходные признаки модели структурно изменены для повышения выразительности модели.

Наконец, чтобы объективно сравнить производительность различных алгоритмов ключевых точек лица, в нашем подходе, в качестве индекса оценки алгоритма используется средняя ошибка нормализации (NME).

Формула расчета выглядит следующим образом:

$$e = \frac{\sum_{i=1}^{N} x_i - x_i^*}{N * d}, \quad (2)$$

где x_i — предсказанная i координата, x_i* — i-й слева буква Ground-Truth, d — евклидово расстояние между двумя координатными точками, а N — количество ключевых точек.

2. Алгоритм обнаружения усталости при вождении

Алгоритм PFLD [5], используемый в авторском проекте, использует 106 ключевых точек лица, как показано на рис. 3, из трех углов лица для обнаружения эффекта PFLD. Можно видеть, что ключевые точки в каждой позиции могут быть расположены более точно, что может быть достаточно точно представить в виде маркировки глаз и рта на лице водителя Функциональный алгоритм обнаружения усталости при вождении, такого как сонливость, зевота и взгляд влево и вправо, осуществляется в таких областях, как контуры тела, рта и лица. Следующие три аспекта будут подробно рассмотрены, см. рисунок 3.

Рис. 3. Пример диаграммы ключевых точек лица с 106 точками на основе PFLD

По статистике, в нормальных условиях частота моргания водителя составляет около десятка раз в минуту, но при утомлении частота моргания значительно увеличивается. Институт Карнеги-Меллона предложил алгоритм PERCLOS, основанный на большом количестве экспериментов, который представляет собой количественную величину для измерения состояния усталости водителя [6].

Исследователи обнаружили, что показатель PERCLOS был наиболее коррелирован с усталостью в 9 экспериментах, связанных с показателями обнаружения усталости. Основной механизм PERCLOS для определения усталости заключается в определении уровня усталости водителя путем расчета доли времени, когда глаза закрыты в определенный период времени. Его можно представить формулой 3:

$$P = \frac{\sum_{i=1}^{T} t_i}{T} \times 100\%, \quad (3)$$

где T представляет фиксированный период времени, а t_i — состояние закрытых глаз определенного временного промежутка между 0 и T. Когда водитель утомлен, время состояния закрытых глаз будет увеличиваться, тогда также будет увеличиваться доля состояния закрытых глаз в единицу времени, а также будет увеличиваться полученный показатель PERCLOS.

Для индикатора PERCLOS обычно используются три стандарта: P70, P80 и EM. P70 означает, что отношение закрытой площади глаза ко всему зрачку составляет более 70%, глаз определяется как закрытый, и подсчитывается доля времени, которое глаз закрыт в единицу времени. Точно так же P80 означает, что глаза закрыты, когда соотношение превышает 80%. EM является более строгим, что означает, что отношение закрытой площади ко всему зрачку - больше половины глаза считается закрытым.

Вышеуказанная единица времени может быть преобразована в количество кадров в практических приложениях, поэтому приведенная выше формула может быть выражена в виде более общей формулы 4:

$$P = \frac{f_1}{f} \times 100\%, \tag{4}$$

где $f1$ представляет собой количество кадров определенного типа усталостного поведения при вождении, а f представляет общее количество кадров во время обнаружения.

В работе предлагается рассчитать степень закрытия глаз на основе ключевых точек глаза (как показано на рис. 4) через показатель EAR – степень закрытия глаза. EAR более эффективен, быстрее и проще в реализации, чем другие методы.

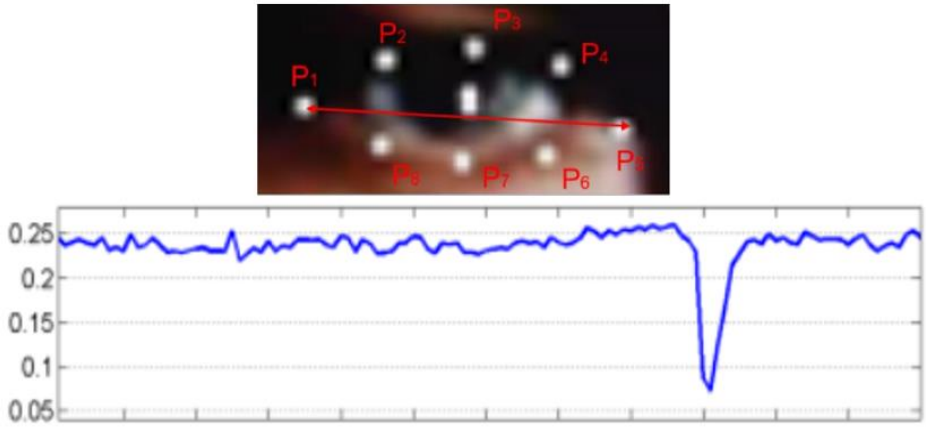


Рис. 4. Ключевые точки глаза и значения показателя степени закрытия глаза

EAR определяется следующим образом, формула 5:

$$E = \frac{\| p_2 - p_8 \| + \| p_3 - p_7 \| + \| p_4 - p_6 \|}{3\| p_1 - p_5 \|}, \tag{5}$$

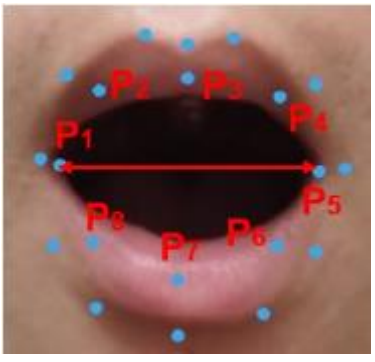


Рис. 5. Ключевые точки рта

где $p1$~$p8$ — это положения ключевых точек глаза, полученные с помощью алгоритма PLFD. Числитель вычисляет расстояние характерных точек глаза в вертикальном направлении, а знаменатель вычисляет расстояние характерных точек глаза в горизонтальном направлении. Поскольку существует только одна группа горизонтальных точек и три группы вертикальных точек, знаменатель умножается на 3, чтобы убедиться, что веса двух групп характерных точек одинаковы. По графику можно оценить, что EAR практически не изменяется при открытых глазах и колеблется вверх и вниз в небольшом диапазоне, однако при закрытых глазах EAR быстро снижается. Это также принцип обнаружения моргания: когда глаз открыт, соотношение сторон глаза почти постоянно, но, когда происходит моргание или закрывание глаза, значение E можно рассчитать по формуле,

чтобы оно было почти нулевым. Следовательно, на основе этого метода расчета можно эффективно количественно оценить степень закрытия глаз.

Зевота также является явным признаком усталости водителя. Количество зевков также значительно увеличивалось, когда водитель утомлен в течение длительного времени. Зевота похожа на закрытие глаз. Таким образом, зевота обнаруживается с помощью метода суждения, аналогичного описанному выше. На рисунке 5 представлены ключевые точки рта. Можно увидеть, что степень открытия и закрытия рта более очевидна, чем у глаз. Ссылаясь на индекс соотношения сторон глаз, степень открытия рта (Mouth Opening Degree, MOD) можно определить по формуле 6:

$$M = \frac{||p_2 - p_8|| + ||p_3 - p_7|| + ||p_4 - p_6||}{3||p_1 - p_5||}, \quad (6)$$

где $p1{\sim}p8$ — это положения ключевых точек рта, извлеченные из алгоритма PLFD. В нормальном состоянии открывание рта практически постоянно, со значением около 0, но при зевоте значение открывания рта значительно увеличивается и намного превышает нормальное открывание рта. Следовательно, этот метод расчета может эффективно определять степень закрытия рта. В дальнейшем данный способ определения состояния усталости будет реализован в приложении, которое найдет свое применение в оценке состояния усталости водителя.

Заключение

В заключении отметим, что представленные в статье модели распознавания усталости водителя реализованы в различных инструментальных библиотеках, таких как:

PyTorch - библиотека машинного обучения с открытым исходным кодом, упрощающая создание и обучение моделей глубокого обучения, в том числе для распознавания лиц и построения 3D-моделей. В ней реализован алгоритм PFLD для расставления ключевых точек на лице.

TensorFlow – это библиотека машинного обучения с открытым исходным кодом, разработанная компанией Google. Она позволяет создавать и обучать различные модели глубокого обучения, включая алгоритм CNN (Convolutional Neural Network).

Литература

1. Петросянц Д. Г. Нейросетевая сверточная модель анализа зрачковых реакций для оценки функционального состояния усталости водителей автотранспортных средств.\ Д. Г. Петросянц и др.\\ ФГБОУ ВО «Казанский национальный исследовательский технический университет им. А. Н. Туполева – КАИ», Казань — 2019. — с. 145–152.
2. Кореневский Н. А. Нечеткие модели оценки уровня эргономики технических систем и её влияния на состояние здоровья человека-оператора с учетом функционального резерва организма. \ Н. А. Кореневский, С. Н Родионова. Т. Н Говорухина. М. А Мясоедова \\ Моделирование, оптимизация и информационные технологии. 2019;7(1). Доступно по: https://moit.vivt.ru/wp-content/uploads/2019/01/KorenevskiySoavtori_1_19_1.pdf
3. Хэ Цзяньбао, Оценка и анализ небезопасного поведения водителя [J], Times Auto, 2021(05): 175–176.
4. Го Биньлин. Анализ небезопасного поведения водителей автомобилей [J] Техническое обслуживание автомобилей и вождения (Maintenance Edition), 2018(06): 80–82.
5. У Ювэй, Ян Синъян, Исследование психологии безопасности водителей скоростных автомагистралей [J], China Science and Technology Information, 2011(21): 87–101.
6. Ван Лянцзюнь, Чен Яньфэй, Тао Юмэн и др. Разработка системы мониторинга состояния вождения для водителей городских поездов [J], Сигналы железнодорожной связи,

2020, 56(10): 79–82.

УДК 51-3

ПРИМЕНЕИЕ ДИНАМИЧЕСКИХ БАЙЕСОВСКИХ СЕТЕЙ ПРИ ОПТИМИЗАЦИИ ПРОЦЕССА ТЕСТИРОВАНИЯ

Д.А. Вервейн

Воронежский государственный университет

Введение

Качественная разработка любого программного обеспечения включает целую серию этапов: разработка требований, проектирование, программирование, тестирование, реализация и ввод в эксплуатацию. Тестирование является важнейшим этапом жизненного цикла разработки программного обеспечения. Тестирование представляет собой непрерывный процесс, сопровождающий каждый этап разработки программного обеспечения. В контексте информационных технологий, в частности, разработки программного обеспечения, «тестирование» определяется как процесс оценивания соответствия фактического и ожидаемого поведения программного обеспечения. Тестирование позволяет вовремя обнаружить несоответствия в работе программного обеспечения и принять решение по доработке проекта, изменению проекта, изменению состава разработчиков и другим актуальным вопросам разработки. Эти процедуры необходимы для того, чтобы QA инженеры и руководители могли предупреждать возникновение ошибок во время непосредственной эксплуатации программного обеспечения в рамках разрабатываемой информационной системы. Исследование закономерностей тестирования однородных групп программного обеспечения могло бы позволить по прохождению определенных этапов тестирования предсказывать с определенной вероятностью статус предстоящих этапов тестирования. Одним из эффективных инструментов обучения и предсказания по имеющимся свидетельствам анализируемого процесса значений определенных концептов данного процесса являются Байесовские сети.

Байесовские сети бывают статические и динамические. Статические байесовские сети способны агрегировать информацию об одном временном срезе тестирования, а динамические байесовские сети доверия способны моделировать серию временных срезов процедуры тестирования. На основе метода вероятностного вывода, разработанного для динамических байесовских сетей, результаты тестирования на предыдущих временных срезах могут использоваться как свидетельства в задачах прогнозирования вероятности возникновения ошибок в будущих предполагаемых этапах тестирования. Моделирование процесса тестирования с помощью динамических байесовских направлено на создание ресурсосберегающих технологий тестирования, способных генерировать информацию о процессе тестирования, обучатся на прецедентах и с определенной вероятностью прогнозировать наличие ошибок или их исправление в определенной версии разрабатываемого программного обеспечения.

1. Определение Байесовских сетей

Процесс тестирования программного обеспечения является сложным стохастическим процессом, имеющим сложную структуру взаимосвязей между подпроцессами и направленным на обнаружение разнородных ошибок. Для моделирования таких процессов необходимы инструменты, которые имеют вероятностную природу, способны обучатся и агрегировать информацию, превращая ее в специальную структуру, удобную для статистического анализа. Одним из таких инструментов, хорошо апробированных в различных прикладных областях, являются байесовские сети. Байесовская сеть доверия представляет собой граф без направленных циклов, то есть ациклический направленный граф, вершинам которого соответствуют случайные величины, а ребра между вершинами соответствуют условным зависимостям между элементами. Каждый случайный концепт описывается функцией распределения вероятности, представленной в виде тензора условных вероятностей.

Пусть X случайный элемент, принимающий значения из множества $\{x_1, \dots, x_n\}$. Вероятность $p(X = x_i)$ будем обозначать как $p(x_i)$. Распределение вероятности X будем обозначать как $P^{(X)}$.

Направленный граф будем обозначать $G^{(V,L)}$., где V – множество вершин $\{v_1, \dots, v_n\}$, а L– множество направленных ребер $\{(u,v)|u,v \in V, u \neq v\}$. Через $\text{pa}^{(v)}$ будем обозначать множество родительских вершин u, для которых существует ребро (u,v), то есть множество вершин, из которых исходит ребро, направленное в v.

Формально, байесовской сетью доверия является пара (G,P) где G – ациклический направленный граф, а P–множество тензоров условных вероятностей $P(X|pa(X))$. Предполагается, что распределение X не зависит от других элементов сети при заданных значениях его потомков и родителей.

Байесовские сети доверия являются вероятностными графическими моделями, к ним применимо правило декомпозиции, обусловленное предположением об условной независимости. Данное правило можно записать в следующем виде: $P\left(X_1, \dots, X_{m)}\right) = P\left(X_1 \middle| pa(X_1)\right) \cdot \dots \cdot P(X_m|pa(X_m))$ где $P\left(X_1, \dots, X_{m)}\right)$ это совместное вероятностное распределение всех случайных концептов модели, $P\left(X_i \middle| pa(X_i)\right)$ это распределение случайного концепта X_i при условии, что известно распределение случайных величин $^{pa(X_i)}$ которые являются родителями X_i, $i = 1, \dots, m$.

Основой для построения байесовской сети доверия могут служить статистические данные или экспертные оценки. Для того чтобы получить оценки условных вероятностей и установить связи между случайными элементами сети необходимо использовать экспертную информацию.

2. Пример применения Байесовских сетей в области тестирования

В качестве примера моделирования процесса тестирования с помощью динамических байесовских сетей, рассмотрим модель тестирования межсайтингового скриптинга методом фаззинга. Организация расширенного пользовательского интерфейса предполагает использование различных скриптов, вставляемых в html страницу. Возможность изменять содержимое на основе действий пользователявыполняется путем обращения к dom-модели страницы. Вызов ошибки межсайтового скриптинга (XSS) представляет собой вставку синтаксических конструкций языка JavaScript во входные параметры с целью их последующего отображения внутри браузера пользователя. Воздействия с помощью вызова данной ошибки направлено на получения критичных данных: сохраненные пароли, идентификаторы сессий и

cookie. Механизмы фрагментирования и трансформации вставляемого кода позволяет осуществить обход фильтров безопасности.

На рисунке 1 приведен фрагмент динамической байесовской сети тестирования уязвимостей межсайтового скриптинга методом фаззинга. Обозначения модели приведены в табл. 1.

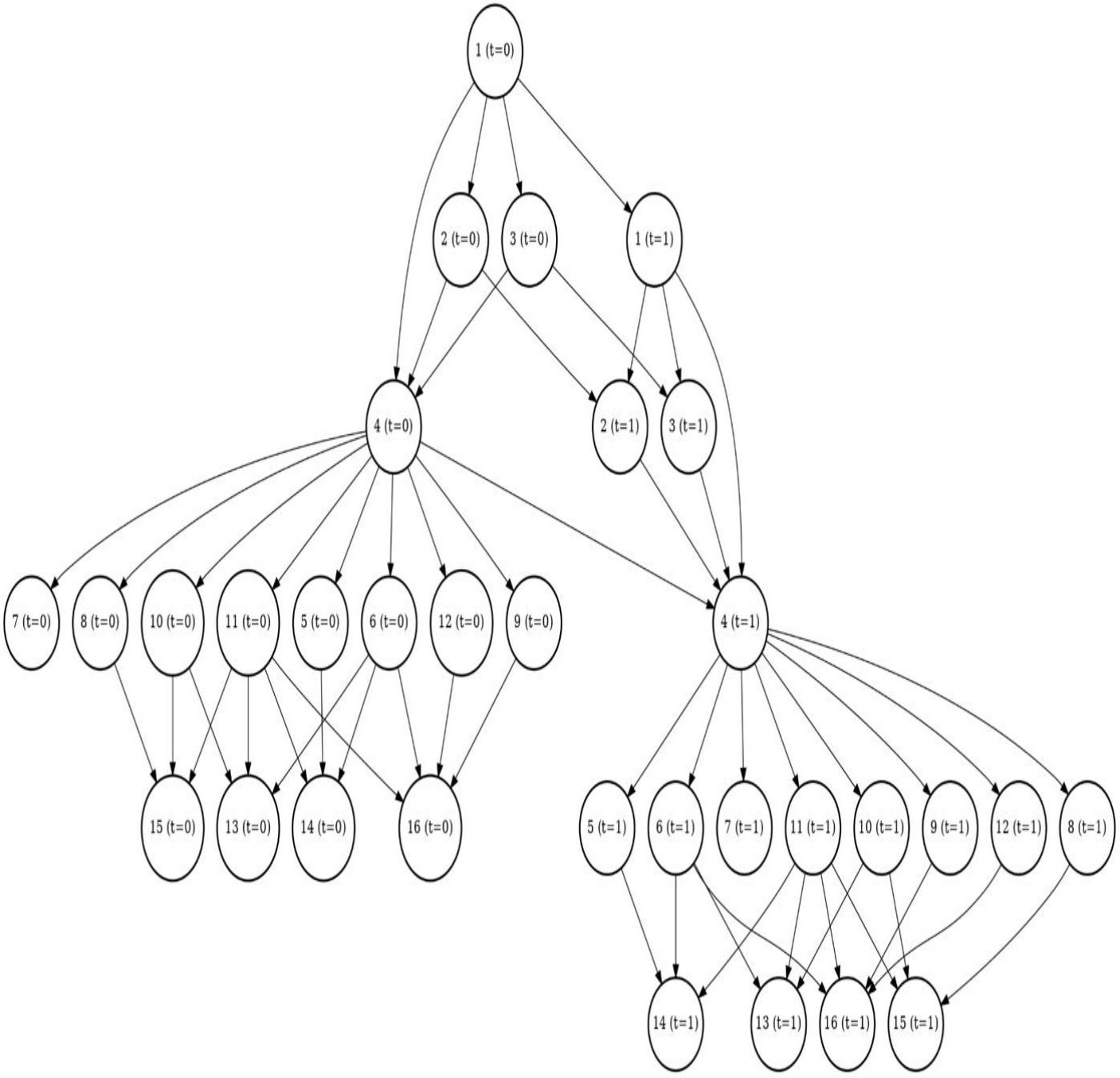


Рис. 1. Два среза динамической байесовской сети тестирования уязвимостей межсайтового скриптинга методом фаззинга

Таблица 1. Описание узлов ДБС фаззинга межсайтового скриптинга

№ п/п	Полное название	Характеристика
1.	XSSType	Определение типа межсайтового скриптинга.
2.	Encoder	Кодирование и преобразование отдельных ключевых слов и параметров, передаваемых в веб-приложение.
3.	Evasion	Механизмы «запутывания» (преобразования) полезной нагрузки.
4.	Xss Payload	Тип используемой полезной нагрузки (html- тэги, обработчики событий)
5.	Keylogger Module	Запоминание комбинаций клавиш, нажатых пользователем.
6.	Spy Eye Module	Получения снимка html-страниц активных вкладок веб-браузера пользователя.
7.	DDos Module	Атака отказа в обслуживании на сторонние ресурсы.
8.	Port Scanner Module	Сканирование открытых портов на компьютере пользователя.
9.	Network Scanner Module	Сканирование локальной сети пользователя.
10.	Nat Pinning Module	Обход сетевых правил маршрутизатора NAT (проникновение в локальную сеть)
11.	Drive By Download Module	Перенаправление пользователя на ресурсы, содержащие вредоносные программы и вирусы.
12.	Browser Fingerprint	Определение типа веб-браузера, списка установленных модулей и компонентов.
13.	Authentication	Аутентификация.
14.	Authorization	Авторизация.
15.	Integrity	Целостность.
16.	Availability	Доступность.
17.	Confidentiality	Конфиденциальность.

В таблице приведем описание вершин, входящих в состав динамической байесовской сети «Межсайтовый скриптинг», характеризующих узлы в соответствии с их функциональным предназначением в процессе решения задач фаззинга. Каждый узел представляет собой генератор тестов, формируемый для тестирования определенного типа ошибок межсайтового скриптинга.

Заключение

Применение динамических байесовских сетей для моделирования процессов тестирования программного обеспечения позволяет придать более четкую структуру процессу

тестирования, отражающую последовательность шагов, предикаты условий, информационные цепочки результатов тестирования и оптимизировать временные и информационные ресурсы процедуры тестирования. Методы моделирования с помощью байесовских сетей позволяют обучать структуру и вероятностные параметры сетей для конкретного процесса тестирования и осуществлять вероятностный вывод для прогнозирования вероятностей наличия ошибок в скрытых (еще не тестируемых) этапах.

Литература

1. Азарнова Т.В. Исследование процесса фаззинга SQL-инъекций веб-приложений на основе динамической сети Байеса / Т.В. Азарнова, П.В. Полухин // Вестник Воронежского государственного ун-та. Серия: Системный анализ и информационные технологии, 2014. - № 1. – С. 120-129.
2. Азарнова Т.В. Применение динамических баейсовских сетей для повышения эффективности процесса фаззинга SQL-инъекций веб-приложений/ Т.В.Азарнова, П.В. Полухин // Системы управления и информационные технологии, 2014. – №1.1(55). – С. 106-112.
3. Азарнова Т.В. Расширение функциональных возможностей фаззинга веб-приложений на основе динамических сетей Байеса / Т.В. Азарнова, П.В. Полухин // Научно-техническая информация. Серия 2. Информ. процессы и системы, 2014. - № 9. – С. 12-19.
4. Алгазинов Э.К. Анализ и компьютерное моделирование информационных процессов и систем / Э. К. Алгазинов, А. А. Сирота [Под общ. ред. А. А. Сироты]. – М.: Диалог-МИФИ, 2009. – 416 с.
5. Арустамов С.А. Применение динамической байесовской сети в системах обнаружения вторжений / С.А. Арустамов, В.Ю. Дайнеко // Научно-технический вестник информационных технологий, механики и оптики, 2012. – №3.- С. 128-133.
6. Бабиков В. Баесовские сети доверия и некоторые аспекты идентификации систем / В. Бабиков. – Спб.: УТЭОСС-2012. – [Электронный ресурс]. – Режим доступа: http: // uteoss2012.ipu.ru/ procdngs/0353.pdf

УДК 519.217

ФОРМИРОВАНИЕ СТРАТЕГИИ УПРАВЛЕНИЕ ЗАПАСАМИ НА ОСНОВЕ МОДЕЛЕЙ ТЕОРИИ МАССОВОГО ОБСЛУЖИВАНИЯ (НА ПРИМЕРЕ АПТЕЧНЫХ ОРГАНИЗАЦИЙ)

М.В.Власова

Воронежский государственный университет

Введение

Управление запасами является важной задачей для любой организации, в том числе и для аптечных. Это связано с тем, что в аптеках продаются лекарства и медицинские препараты, которые имеют ограниченный срок годности и подвержены риску порчи и устаревания. Поэтому, эффективное управление запасами может существенно повлиять на прибыльность аптеки и качество обслуживания пациентов.

Одним из ключевых элементов управления запасами является оптимизация уровня запасов. Слишком большое количество запасов может привести к излишним затратам на хранение, а также к риску устаревания товаров [1]. С другой стороны, недостаток запасов может привести к упущению выгодных продаж и потере клиентов.

Кроме того, для эффективного управления запасами в аптечных организациях необходимо учитывать ряд специфических факторов. Во-первых, необходимо следить за сроком годности товаров и регулярно проверять их состояние. Во-вторых, необходимо учитывать спрос на товары, анализировать статистику продаж и учитывать сезонные колебания спроса. В-третьих, необходимо учитывать стоимость товаров и устанавливать оптимальную цену продажи.

1. Методы *теории массового обслуживания в управлении запасами*

Вероятностные модели теории запасов очень похожи на модели теории массового обслуживания. В данных теориях используются случайные потоки, такие как пуассоновские, составные пуассоновские и рекуррентные. В теории запасов время выполнения заказа на пополнение запасов является аналогом времени обслуживания в теории массового обслуживания (ТМО).

Некоторые модели теории запасов можно представить, как модели теории массового обслуживания с коллективным обслуживанием запросов. Аппарат стохастического анализа моделей теории запасов также связывает эту теорию с теорией массового обслуживания, в основном через цепи Маркова, регенерирующие и полурегенерирующие процессы. Одним из основных признаков ряда моделей теории запасов является возможность свести задачу анализа к изучению одинаковых процессов в периоды регенерации, что значительно упрощает задачу анализа.

В теории массового обслуживания главной задачей является стохастический анализ, а в теории запасов первое место имеет задача нахождения оптимальных решений. В общей теории оптимизации модели управления запасами представляются как частный случай [3].

В качестве примера возможного применения моделей управления запасами при наличии стохастических процессов будем использовать аптечные организации. Когда возникает скачок заболеваний, аптеки сталкиваются с рядом проблем: задержками в поставке товаров, отклонением фактического спроса от ожидаемого, дефицитом отдельных лекарственных препаратов и списанием лекарств из-за истечения срока годности. Все эти факторы приводят к дополнительным затратам для аптечных организаций.

Чтобы учесть вероятностный характер работы аптек и случайный спрос на лекарственные средства в данный период, в качестве стратегии управления запасами можно использовать модель ТМО.

2. Модель управления запасами с фиксированным размером заказа

Предположим, что покупатели приходят в аптечную организацию при интенсивности $\boldsymbol{\mu}$ и имеет пуссоновское распределение. То есть в аптеку посетители приходят в разное время, при этом вероятность того, что один покупатель придет в конкретном интервале времени от $\mathbf{t}$ до $\mathbf{t + \Delta t}$ равна $\boldsymbol{\mu\Delta t}$ и не зависит от $\mathbf{t}$, но при этом вероятность того, что два и более посетителя придут в одном промежутке времени очень маленькая. Такое утверждение оказывается абсолютно подтвержденным, если $\boldsymbol{\Delta t}$ будет маленьким числом. Лекарственные средства (ЛС) доставляют в аптеку с интенсивностью $\boldsymbol{\lambda}$ и имеют пуассоновские распределения. Время с начала создание заявки для пополнения резервов до времени, когда в аптеку прибудут лекарственные средства, имеет показательное распределение. В законе пуассоновского потока событий время поступления заказа имеет вид $\mathbf{T = 1/\lambda}$. Считаем, что $\boldsymbol{\lambda}$ и $\boldsymbol{\mu}$ являются определенными и не зависят от времени управления. Будем считать, что, когда количество

запасов лекарственных средств понижается до низкого(критического) уровня **P**, который назовем «точкой заказа», то мы будем заказывать **Q** единиц лекарственных средств, что:

P+Q=M; (1)

P-точка заказа;

Q-количество лекарственных единиц, которое необходимо заказать;

M – максимальное количество лекарственных средств, которое принимает аптека.

Величина **M** является заданной. Отсюда определению подходит либо точка заказа **P**, либо объём заказа **Q** для уникального лекарственного средства. При определении одного из этих значений, второе находим из уравнения (1).

Sn обозначаем, как состояние системы массового обслуживания (СМО), при условии, что СМО будет находиться в виде **Sn**, когда будет присутствовать **n** единиц лекарственных средств в аптеке. Система покинет состояние **Sn** и перейдет в **Sn+1**, когда аптека получит новую единицу лекарственного средства, или в состояние **Sn–1**, когда покупатель купил единицу лекарственного средства. В случае применение стратегии «при заказе **Q** единиц лекарственных средств, когда уровень запасов уменьшился до **P**, и заказе **M** единиц лекарственных средств, когда уровень запасов равен нулю», это значит, что:

1. система массового обслуживания перешла из состояния **Sn** в состояние **Sn–1** когда была продана одна единица лекарственного средства с интенсивностью μ;

2. переход системы из состояния **Sn (n ≠ 0)** в состояние **Sn+Q**, а из состояния **S0** в состояние **SM**, при доставке запасов с интенсивностью **λ**. [3]

Схема случайного процесса, например, для системы S с шестью состояниями (**M** равно пяти, а **P** двум единицам лекарственных средств) показана на рис. 1.

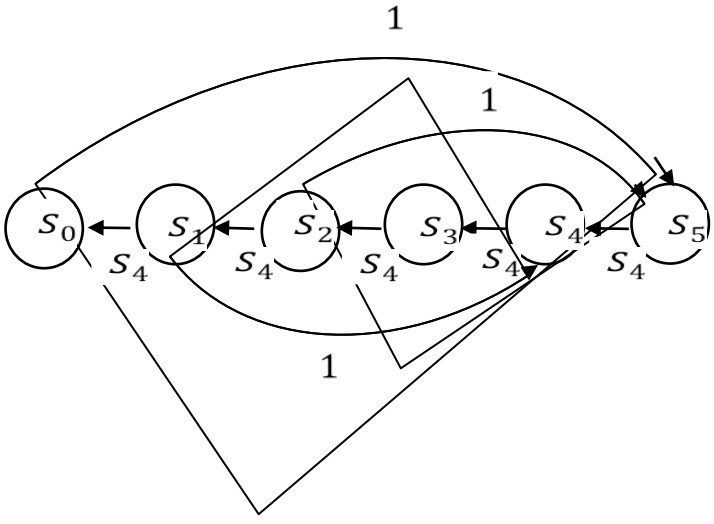


Рис.1. Схема случайного процесса движения запасов

На рис. 1 показано, что когда запасов нет на складе, то формируется заказ на доставку запасов до величины M, то есть пяти единиц. Когда лекарственные средства находятся в состояние малого количества и их сумма равна двум единицам, то формируется заказ лекарственных средств в размере трёх единиц. Доставить сформированный заказ на лекарственные средства могут:

1. до прибытия клиента, и тогда система перейдёт из состояния **S2** в состояние **S5**;
2. после прибытия клиента и продажи ему единицы лекарственных средств, и тогда система перейдёт из состояния **S1** в состояние **S4** и т.д. [2]

Pn обозначим, как вероятность того, что в наличии есть n единиц лекарственных средств, а через **dPn/dt** – скорость изменения состояния **Sn**. Тогда скорость изменения размеров запаса товара можно определить с помощью уравнений Колмогорова:

$$\frac{dp_0}{dt} = -\lambda p_0 + \mu p_1; \quad (2)$$

$$\frac{dp_1}{dt} = -\lambda p_1 - \mu p_1 + \mu p_2; \quad (3)$$

$$\frac{dp_2}{dt} = -\lambda p_1 - \mu p_1 + \mu p_3; \quad (4)$$

$$\frac{dp_3}{dt} = -\mu p_3 + \mu p_4; \quad (5)$$

$$\frac{dp_4}{dt} = \lambda p_1 - \mu p_4 + \mu p_5; \quad (6)$$

$$\frac{dp_5}{dt} = \lambda p_0 + \lambda p_2 - \mu p_5. \quad (7)$$

После решения всех уравнений мы можем рассчитать прибыль как:

$$F = (\vartheta - \nu)u(1 - p_0) - c_1\frac{\mu}{Q} - C_2\bar{n} \quad (8)$$

Итоговое уравнение для нахождения максимального объема партии:

$$F = (\vartheta - \nu)\mu\left(1 - \frac{\mu^{p+1}}{(\mu + \lambda, P[\mu + (M - P)\lambda]]}\right) - C_1\frac{\mu}{M - p} - \quad C_2\left(\frac{\mu^{p+1}}{(\mu + \lambda, P[\mu + (M - P)\lambda]]u}\right) *$$

$$* \left[\frac{1 - \left(\frac{\mu+\lambda}{\mu}\right)^p\left(p + 1 - p\frac{\mu+\lambda}{\mu}\right)}{\left[1 - \frac{\mu+\lambda}{\mu}\right]^2} + \left(\frac{M(M+1) - P(P-1)}{2}\right)\left(\frac{\mu+\lambda}{\mu}\right)^p\right] \quad (9)$$

Где ϑ – розничная цена;

ν - закупочная цена лекарственных средств;

μ – интенсивность прибытие покупателей;

M – максимальный объем лекарственного средства;

P – критический уровень запаса;

λ – интенсивность прибытие партии лекарственного средства;

C_1 – затраты на оформление и выполнение заказов;

C_2 – затраты на хранение единицы лекарственного средства.

Заключение

Изложенные в статье методы массового обслуживания могут быть использованы для решения различных задач, связанных с управлением запасами. Эти модели позволяют не только оптимизировать ассортимент и стратегии пополнения запасов, но и проводить моделирование системы обслуживания, оценивать прибыль, используя различные стоимостные показатели и учитывая интенсивность спроса и пополнения.

Литература

1. Григорьев М.Н. Механизмы закупочной логистики // Григорьев М.Н. Логистика: Учебное пособие. - 2-е изд., испр. и доп. - 3.1. - М.: Гардарики, 2007. - 475 с.

2. Истомин А.Л. Модель управления запасами с фиксированным размером заказа/ А.Л.Истомин. А.В.Баденников. А.А.Истомина // Логистика запасов. -2018. - №4.- С. 49-54.

3. Рубальский Г.Б. Стохастическая теория управления запасами / Г.Б.Рубальский // Автоматика и телемеханика – 2009. - № 12 -С. 175-186.

УДК 004.94

ПРИМЕНЕНИЕ ГОЛОСОВОГО УПРАВЛЕНИЯМИ ПЕРСОНАЖАМИ В UNITY3D

А. Д. Волгин, Е.В. Трофименко

Воронежский государственный университет

Введение

В наше время все чаще в компьютерных играх применяется голосовое управление персонажами. В данной статье приводится обзор алгоритмов по распознаванию речи. Также рассматривается подключения модуля голосового управления персонажем к Unity3D при разработке игр

1. Обзор основных видов алгоритмов по распознаванию речи

Алгоритм распознавания речи работает следующим образом. Сначала устройство записывает голосовой запрос, а нейросеть анализирует поток речи. Волна звука делится на фрагменты — фонемы.

Затем нейросеть обращается к своим шаблонам и сопоставляет фонемы с буквой, слогом или словом. Далее образуется порядок из известной программы слов, а неизвестные слова она вставляет по контексту. В результате объединения информации с этих двух этапов получается перевод речи в текст. На заре развития процесс работы Speech-to-Text заключался в элементарной акустической модели речь человека сопоставлялась с шаблонами. Но количества словарей в системе было недостаточно для точного распознавания, программа часто ошибалась. Современные системы распознавания речи имеют следующие основные модули: акустическая модель, языковая модель, декодер

1.1 Акустические модель

Акустическая модель - это функция, принимающая на вход признаки на небольшом участке акустического сигнала (фрейме) и выдающая распределение вероятностей различных фонем на этом фрейме. Таким образом, акустическая модель дает возможность по звуку восстановить, что было произнесено с той или иной степенью уверенности [1]. Пример акустическая модель для слова six представлен на рисунке 1.

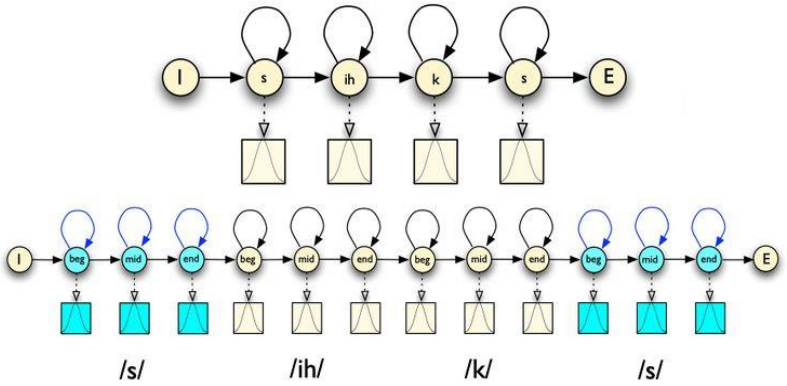


Рис. 1 Акустическая модель для слова six

В круглых состояниях изображены фонемы, а в квадратных, распределения вероятностей признаков. Фонемы часто разбивают на 3 этапа — начало, середину и конец, потому что фонема может звучать по-разному в зависимости от момента времени её произнесения. Каждое скрытое состояние содержит переход само в себя, так как время произнесения одной фонемы может занять несколько фреймов.

1.2 Языковая модель

Языковая модель — позволяет узнать, какие последовательности слов в языке более вероятны, а какие менее. Здесь в самом простом случае требуется предсказать следующее слово по известным предыдущим словам. В традиционных системах применялись модели типа N-грамм, в которых на основе большого количества текстов оценивались распределения вероятности появления слова в зависимости от N предшествующих слов. Для получения надежных оценок распределений параметр N должен быть достаточно мал: одно, два или три слова — модели униграмм, биграмм или триграмм соответственно. Внедрение языковой модели в систему распознавания речи позволило значительно повысить качество распознавания за счет учета контекста [1].

1.3 Декодер

В ходе работы системы автоматического распознавания речи задача распознавания сводится к определению наиболее вероятной последовательности слов, соответствующих содержанию речевого сигнала. Наиболее вероятный кандидат должен определяться с учетом как акустической, так и лингвистической информации.

Это означает, что необходимо производить эффективный поиск среди возможных кандидатов с учетом различной вероятностной информации. При распознавании слитной речи число таких кандидатов огромно, и даже использование самых простых моделей приводит к серьезным проблемам, связанным с быстродействием и памятью систем.

Как результат, эта задача выносится в отдельный модуль системы автоматического распознавания речи, называемый декодером. Декодер должен определять наиболее грамматически вероятную гипотезу для неизвестного высказывания – то есть определять наиболее вероятный путь по сети распознавания, состоящей из моделей слов [1].

2 Реализация

В данной статье рассматривается подключения открытого модуля CMU Sphinx (pocketsphinx) для распознавания команд к Unity3D [2,5]. CMU Sphinx может работать в офлайн режиме, а также, не смотря на не очень высокую точность распознавания, обладает

одной из лучших скоростей распознавания, большим количеством поддерживаемых языков, распространяется под лицензией BSD, которая разрешает встраивание в коммерческие проекты, а также возможностью внесения изменений в открытый код [3]. Имеет высокие показатели WER=21.4% - 22.7%, WRR = 78.6% - 77.3%, где WER – это точность распознавания, которая является показателем качества и определяется как процент неправильно распознанных слов (Word Error Rate), а показатель WRR (Word Recognition Rate) наоборот отражает процент правильно распознанных слов.

Pocketsphinx — это программа, принимающая некие акустические модели, грамматику и словарь, а также звуковой поток/звуковой файл/поток с микрофона, и выдающая распознанный текстовый файл. Подключение pocketsphinx происходит с помощью добавление библиотеки в unity.

Sphinx представляет собой модульный фреймворк (рис. 2). Модульная структура позволяет варьировать параметры системы исходя из требований конкретной задачи. Выделяются 3 основных модуля: FrontEnd, Decoder и Linguist [4]. Модуль FrontEnd преобразует входные данные в вектор параметров. Модуль Linguist на основе выбранных языковой, акустической моделей и словаря строит SearchGraph. Наконец, подмодуль Decoder'а – SearchManager – использует вектор параметров и построенный граф для декодирования и выдаёт результат.

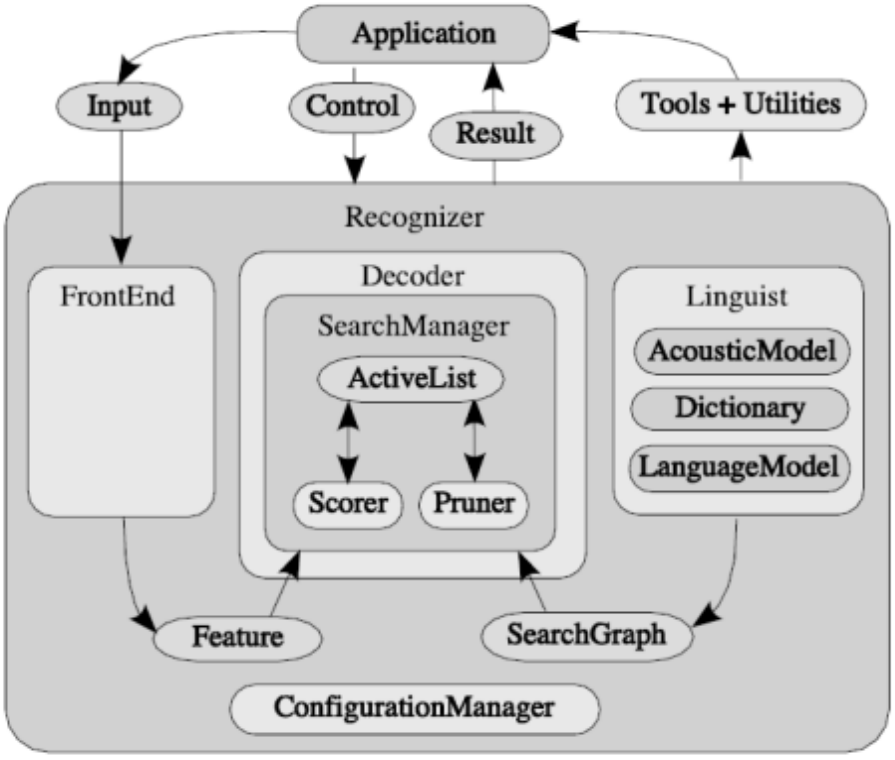


Рис. 2. Взаимодействие модулей системы CMUSphinx

Построение SearchGraph'а в Linguist происходит на основании данных о языке, получаемых из языковой и акустической модели. Каждая из них представляет собой HMM для элементарных звуковых единиц, используемых в конкретной системе. Словарь сопоставляет слова из языковой модели и комбинации элементов акустической модели.

Decoder посылает подмодулю под названием SearchManager запрос на распознавание множества фреймов с указанными выше данными. На каждом этапе работы SearchManager строит все пути, которые достигают конечного непроизводящего состояния. SearchManager

использует алгоритм передачи токенов. Для используемого алгоритма SearchManager может, но не обязан, содержать множество активных токенов (ActiveList). Для упрощения вычислений подмодулем Pruner проводится сокращение множества токенов. Подмодуль Scorer по запросу вычисляет оценки плотности распределения для данных состояний в данные моменты времени. Система CMUSphinx позволяет изменять код любого из модулей, если это необходимо в рамках определённых данных или задачи. Также встроенные средства позволяют адаптировать акустическую модель под речевые особенности конкретных говорящих: акценты, нарушения произношения и т.п.

Пошаговое подключение модуля CMU Sphinx осуществляется по следующему алгоритму:

1. Подключить библиотеку pocketsphinx адаптированную к Unity3D.
2. Подключить SphinxManager который позволяет подключить определенный язык и настроить максимальное и минимальное время прослушивания микрофоном в unity (рис. 3).
3. Подключить скрипт к объекту, который принимает слова. И указать команды и их транскрипцию для нахождения и выполнения команд. На рисунке 4, приведен скрипт с внесенными словами [6,7].

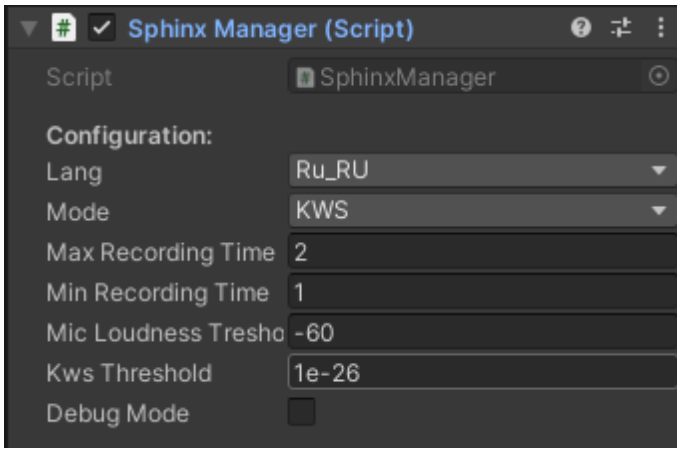


Рис. 3. Подключение и настройка SphinxManager

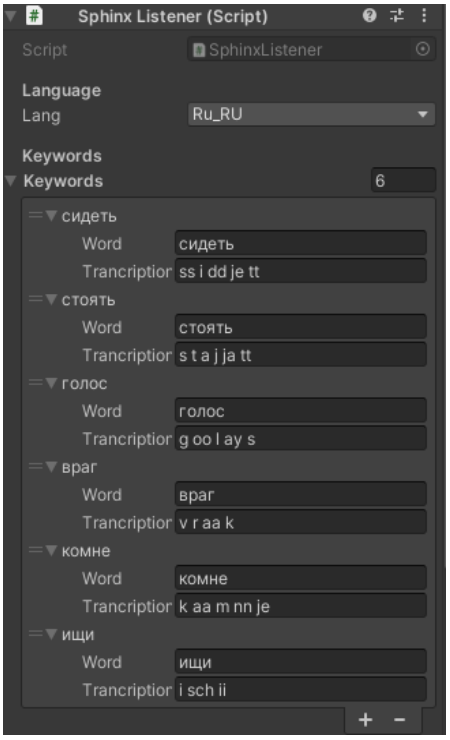


Рис. 4. Подключение и настройка SphinxManager к объекту

Есть скрипт поведения сущности, которая с помощью определенных команд выполняет то или иное действие (рис.5).

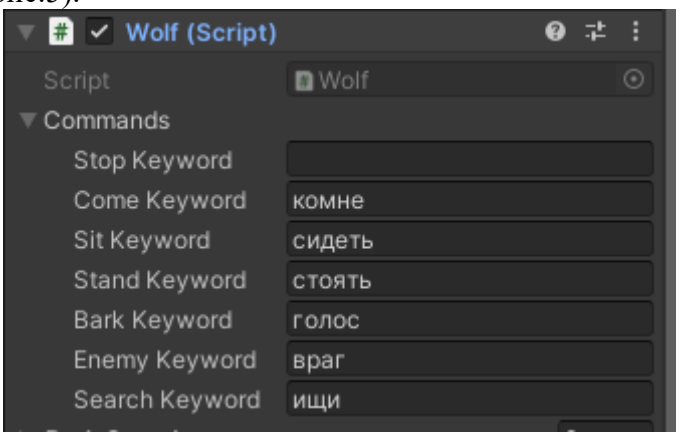


Рис. 5. Подключение и настройка Wolf скрипта

Как итог работы скрипт прослушивает определенную команду, далее сверяет транскрипцию по заданному списку. И если слово похоже, то по алгоритму объект выполняет определенную команду.

Заключение

Были рассмотрены алгоритмы по распознаванию речи и описано подключение открытого модуля CMU Sphinx（pocketsphinx) к Unity 3D.

Литература

1. Тампель И.Б, Карпов А.А. Автоматическое распознавание речи. — СПб. : Университет ИТМО, 2016. — С. 113.
2. CMU Sphinx Project by Carnegie Mellon University – URL: http://cmusphinx.sourceforge.net/– (Дата обращения 25.03.2023)

3. Карпов А. А., Кипяткова И. С. Методология оценивания работы систем автоматического распознавания речи // Известия высших учебных заведений. Приборостроение. – 2012. – Т. 55, №. 11. – С. 38-43.
4. Walker, Lamere, Kwok, Raj, Singh, Gouvea, Wolf, Woelfel. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. 2004
5. PocketSphinxUnityDemo – URL: : https://github.com/Funnyguy77/PocketSphinxUnityDemo (Дата обращения 21.03.2023)
6. Unity – URL: https://unity.com/ru – (Дата обращения 20.03.2023)
7. UnityManual – URL: https://docs.unity3d.com/ru/530/Manual/UnityManual.html (Дата обращения 23.03.2023)

УДК 517.977.52

ДИНАМИКА РАЗВИТИЯ ФИРМЫ

Ю. Д. Воронова

Воронежский государственный университет

Введение

Экономико-математические модели динамики деятельности предприятий помогают изучению деятельности конкретного предприятия, а также с их помощью можно прогнозировать различные показатели в динамике. В работе проведён анализ динамики развития фирмы с управлением. Целью работы является выявления лучшей стратегии выплаты заработной платы при наличии определённого стартового капитала.

1. Постановка задачи

Рассмотрим экономическую модель развития фирмы, где скорость изменения капитала фирмы *x(t)* в момент времени *t* подчиняется уравнению:

$$x' = \psi(x) - \alpha x - p(t,x,u), \quad x(0) = x_0$$

x_0 – начальный капитал фирмы, α – инфляционный фактор (обесценивание денег, износ оборудования и т.д.), $p(t,x,u)$ – зарплата сотрудников, $u(t)$ – управление, $\psi(x)$ – функция полезности. Будем считать, что затраты на оплату труда сотрудникам фирмы изменяются по закону:

$$p(t,x,u)=\frac{x_0}{k}\left(\frac{x}{x_0}\right)^{u(t)}$$

где управление $u(t)$ изменяющееся в промежутке [0,1].

Функцией полезности называется скорость притока суммарной прибыли при вложении капитала x. Функция полезности $\psi(x)$ обычно предполагаются – непрерывная, вогнутая, удовлетворяющая условиям:

$$\psi(0)=0,\psi'(0)=+\infty,\lim_{x\to\infty}\psi(x)=\infty.$$

Вогнутость функции следует из того, что, как правило, размер прибыли не пропорционален вложениям. То есть при увеличение в 2 раза капитала фирмы, мы получим рост прибыли меньше, чем в 2 раза. Происходит это по многим причинам: увеличение издержек, ужесточение конкуренции и т.д. Рассмотрим функцию полезности вида $\psi(x)=\sqrt{x}$, она удовлетворяет всем указанным выше условиям.

Рассмотрим вопрос выработки оптимальной стратегии формирования фонда оплаты труда сотрудников фирмы, которая максимизирует капитал фирмы за некоторый промежуток времени.

Оптимизационная модель может быть представлена в виде задачи оптимального управления со свободным правым концом:

$$J(u)=\int_0^T x(t)dt\to\max \tag{1}$$

при ограничениях:

$$x'=\sqrt{x(t)}-\alpha x(t)-\frac{x_0}{k}\left(\frac{x(t)}{x_0}\right)^{u(t)}, \tag{2}$$

$$x(0)=x_0,0\le u(t)\le 1. \tag{3}$$

2. Применение принципа максимума Понтрягина

Для решения поставленной задачи (1)-(2) применим принцип максимума Понтрягина. Пусть W – множество допустимых процессов в задаче (1)-(3) не пусто и существует локально оптимальный процесс $\tilde{w}=(\tilde{x}(\cdot),\tilde{u}(\cdot))$.

Выпишем функцию Понтрягина в задаче (1)-(3)

$$H(t,x,u,\lambda_0,\lambda(t))=\lambda_0 x(t)+\lambda(t)\left\{\sqrt{x(t)}-\alpha x(t)-\frac{x_0}{k}\left(\frac{x(t)}{x_0}\right)^{u(t)}\right\}, \tag{4}$$

где $\lambda(t)$ – абсолютно непрерывная функция.

Для процесса $(\tilde{x}(\cdot),\tilde{u}(\cdot))$ выполнены следующие условия:

1) условие максимума

$$H(t,\tilde{x}(t),\tilde{u}(t),\lambda_0,\lambda(t))=\max_{0\le u\le 1}H(t,\tilde{x}(t),u(t),\lambda_0,\lambda(t))=$$

$$=\lambda_0\tilde{x}(t)+\lambda(t)\left\{\sqrt{\tilde{x}(t)}-\alpha\tilde{x}(t)\right\}+\frac{x_0}{k}\max_{0\le u\le 1}\left[-\lambda(t)\left\{\left(\frac{\tilde{x}(t)}{x_0}\right)^{u(t)}\right\}\right]. \tag{5}$$

2) сопряженная функция $\lambda(t)$ удовлетворяет сопряженному дифференциальному уравнению

$$\lambda'(t) = -\lambda_0 - \lambda(t)\left\{\frac{1}{2\sqrt{\tilde{x}(t)}} - \alpha - \frac{x_0}{k}\tilde{u}(t)\left(\frac{\tilde{x}(t)}{x_0}\right)^{\tilde{u}(t)-1}\frac{1}{x_0}\right\}$$

3) для правого свободного конца выполнено условие трансверсальности

$$\lambda(T) = 0$$

Так как мы рассматриваем только $\tilde{x}(t) > 0$, то $\left(\frac{\tilde{x}(t)}{x_0}\right)^{\tilde{u}(t)} > 0$ при любом значении

$$0 \le u(t) \le 1.$$

Тогда максимизирующее управление в соотношении (5) имеет вид

$$\tilde{u}(t) = \begin{cases} 1, \textit{если } \lambda(t)\ln\dfrac{\tilde{x}(t)}{x_0} < 0, \\ 0, \textit{если } \lambda(t)\ln\dfrac{\tilde{x}(t)}{x_0} > 0, \\ [0,1], \textit{если } \tilde{x}(t) \equiv x_{0.} \end{cases} \tag{6}$$

Краевая задача принципа максимума запишется в виде

$$\begin{cases} x' = \sqrt{x(t)} - \alpha x(t) - \dfrac{x_0}{k}\left(\dfrac{x(t)}{x_0}\right)^{\tilde{u}(t)} \\ \lambda'(t) = -\lambda_0 + \lambda(t)\left\{\dfrac{1}{2\sqrt{x(t)}} - \alpha - \dfrac{\tilde{u}(t)}{k}\left(\dfrac{x(t)}{x_0}\right)^{\tilde{u}(t)-1}\right\} \\ x(0) = x_0 \\ \lambda(T) = 0 \end{cases} \tag{7}$$

где $\tilde{u}(t)$ определяется по формуле (6).

Исследуем регулярные решения. Пусть $\lambda_0 = 1$. Рассмотрим возможные различные способы задания управления на отрезке $[0,T]$.

2.1. Стратегия выплаты заработной платы с управлением равным 1

Пусть $u(t) = 1, t \in [0,T]$. Тогда необходимо найти решение задачи Коши вида:

$$x' = \sqrt{x} - \left(\alpha + \frac{1}{k}\right)x \tag{8}$$

$$x(0) = x_0$$

Решение задачи Коши (8) имеет вид:

$$x(t)=\frac{e^{-\left(\alpha+\frac{1}{k}\right)t}}{\left(\alpha+\frac{1}{k}\right)^2}\left(e^{\frac{\left(\alpha+\frac{1}{k}\right)t}{2}}-1+\left(\alpha+\frac{1}{k}\right)\sqrt{x_0}\right)^2 \tag{9}$$

Сопряженное дифференциальное уравнение принимает вид:

$$\lambda'=-1-\lambda(t)\left\{\frac{1}{2\sqrt{x(t)}}-\alpha-\frac{1}{k}\right\} \tag{10}$$

Решение уравнения (10) с учетом условия $\lambda(T)=0$ описывается формулой

$$\lambda(t)=\frac{1}{e^{0{,}5\left(\alpha+\frac{1}{k}\right)t}-\left(1-\left(\alpha+\frac{1}{k}\right)\sqrt{x_0}\right)}\left[\left(\alpha+\frac{1}{k}\right)\sqrt{x_0}-1-\frac{e^{0{,}5\left(\alpha+\frac{1}{k}\right)t}}{0{,}5\left(\alpha+\frac{1}{k}\right)-1}+\frac{e^{\left(0{,}5\left(\alpha+\frac{1}{k}\right)-1\right)T+t}}{0{,}5\left(\alpha+\frac{1}{k}\right)-1}-\right.$$
$$\left.-\left(\left(\alpha+\frac{1}{k}\right)\sqrt{x_0}-1\right)e^{-T+t}\right] \tag{11}$$

Отметим, что решение краевой задачи (7) вида (9), (11) возможно только, если условие $\lambda(t)\ln\frac{\tilde{x}(t)}{x_0}<0$ выполнено при всех t из отрезка $[0,T]$. Знак $\ln\frac{\tilde{x}(t)}{x_0}$ согласно исследованиям, будет зависеть от соотношения коэффициентов α, $\frac{1}{k}$ и начального капитала фирмы x_0. А именно,

a) если $x_0<\frac{1}{\left(\alpha+\frac{1}{k}\right)^2}$, то компания будет богатеть, т.е. $\tilde{x}(t)>x_0$ и $\ln\frac{\tilde{x}(t)}{x_0}>0$.

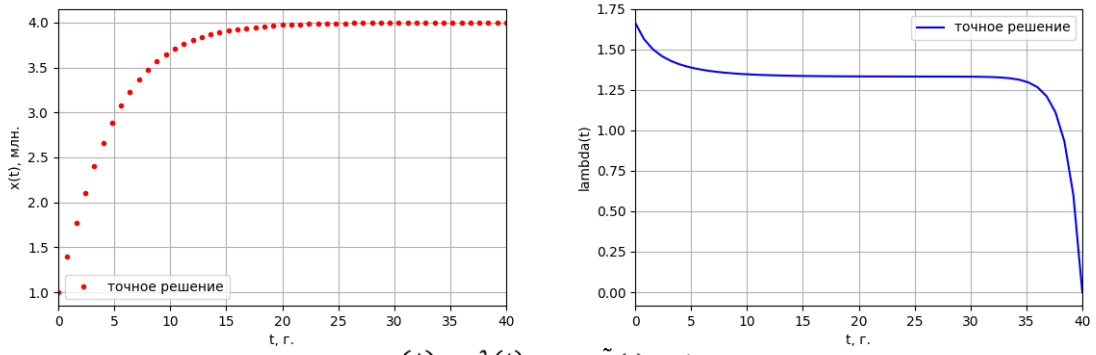


Рис 1. Графики $x(t)$ *и* $\lambda(t)$ *при* $\tilde{u}(t)=1$, *α=0.1, k=2,5 и* x_0*=1*

b) если $x_0>\frac{1}{\left(\alpha+\frac{1}{k}\right)^2}$, то компания будет беднеть, т.е. $\tilde{x}(t)<x_0$ и $\ln\frac{\tilde{x}(t)}{x_0}<0$.

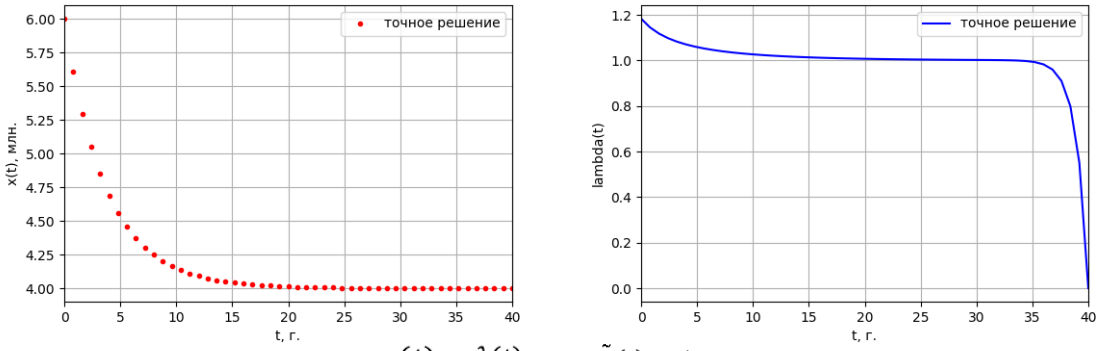


Рис 2. Графики $x(t)$ *и* $\lambda(t)$ *при* $\tilde{u}(t)=1$*, α=0.1, k=2,5 и* x_0*=6*

Вывод: стратегию управления $\tilde{u}(t)=1, t\in[0,T]$, необходимо выбирать, если коэффициенты α, $\frac{1}{k}$ и начальный капитал фирмы x_0 связаны соотношением $x_0 > \frac{1}{\left(\alpha+\frac{1}{k}\right)^2}$.

2.2. Стратегия выплаты заработной платы с управлением равным 0

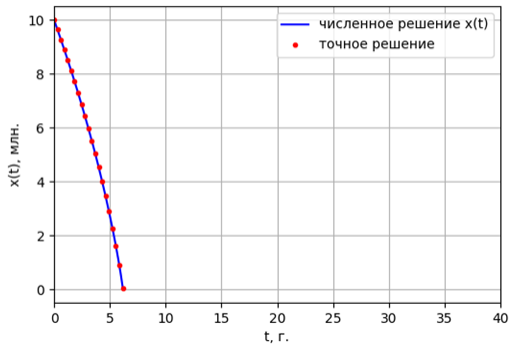
Пусть $\tilde{u}(t)=0, t\in[0,T]$. Тогда необходимо найти решение краевой задачи:

$$\begin{cases} x' = \sqrt{x(t)} - \alpha x(t) - \dfrac{x_0}{k} \\ \lambda'(t) = -\lambda_0 + \lambda(t)\left\{\dfrac{1}{2\sqrt{x(t)}} - \alpha\right\} \\ x(0) = x_0 \\ \lambda(T) = 0 \end{cases} \tag{12}$$

Получить аналитическое решение задачи (12) не представляется возможным, поэтому будем решать её численными методами.

Отметим, что мы будем решать краевую задачу вида (12) только, если выполнено условие $\lambda(t)\ln\frac{\tilde{x}(t)}{x_0} > 0$ при всех t из отрезка $[0,T]$. Знак $\ln\frac{\tilde{x}(t)}{x_0}$ будет зависеть от соотношения коэффициентов α, $\frac{1}{k}$ и начального капитала фирмы x_0. А именно,

а) $\frac{x_0}{k} > \frac{1}{4\alpha}$. Фирма разорится за конечное время.



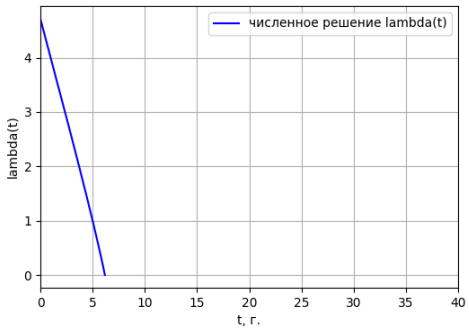
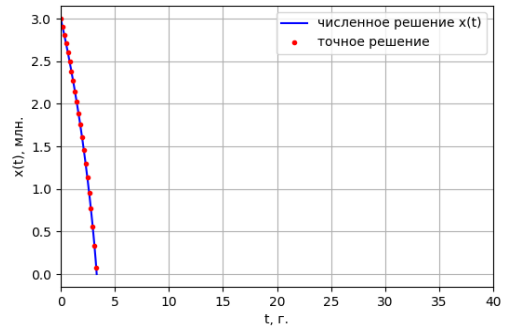


Рис3.Графики $x(t)$ *и* $\lambda(t)$ *при* $\tilde{u}(t) = 0$, *α=0.1, k=3 и* x_0*=10*

b) $\frac{x_0}{k} \le \frac{1}{4\alpha}$ и $x_0 < \left(\frac{1-\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2$, фирма разоряется за конечное время.



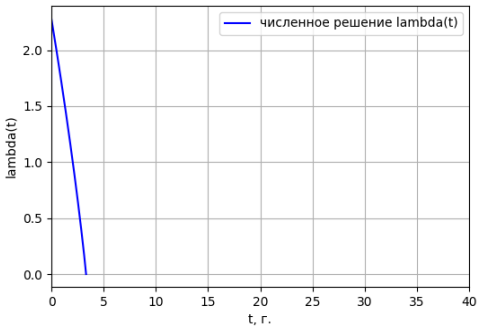


Рис 4.Графики $x(t)$ *и* $\lambda(t)$ *при* $\tilde{u}(t) = 0$, *α=0.1, k=1,5 и* x_0*=3*

c) $\frac{x_0}{k} \le \frac{1}{4\alpha}, x_0 \in \left(\left(\frac{1-\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2, \left(\frac{1+\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2 \right)$ решение x(t) монотонно возрастает и стремится к $\left(\frac{1+\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2$ при $t \to \infty$.

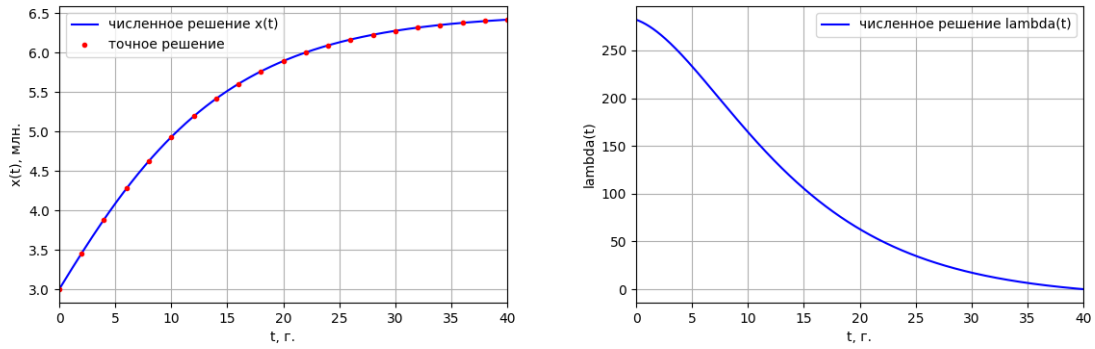


Рис 5.Графики $x(t)$ *и* $\lambda(t)$ *при* $\tilde{u}(t)=0$*, α=0.3, k=5 и* $x_0=3$

d) $\frac{x_0}{k} \le \frac{1}{4\alpha}$ и $x_0 > \left(\frac{1+\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2$ капитал фирмы экспоненциально падает со временем и выходит на «потолок» $\left(\frac{1+\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2$.

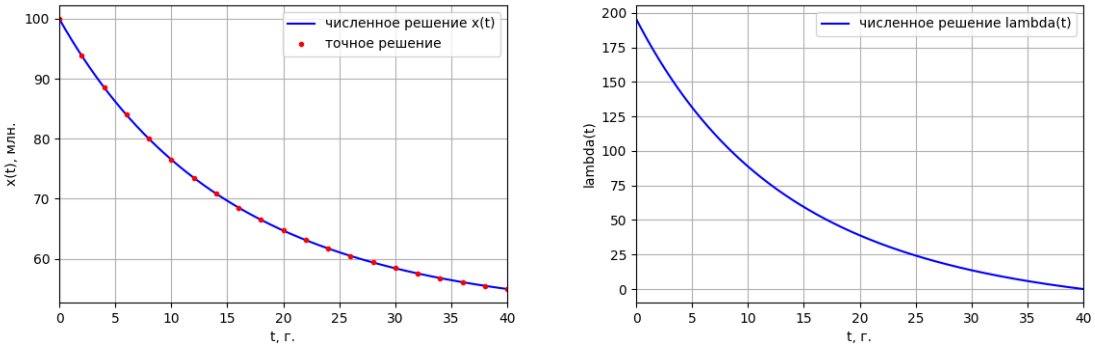


Рис 4.Графики $x(t)$ *и* $\lambda(t)$ *при* $\tilde{u}(t)=0$*, α=0.125, k=125 и* $x_0=100$

Вывод: стратегию управления $\tilde{u}(t)=0, t\in[0,T]$, выгодно выбирать, если коэффициенты a, $\frac{1}{k}$ и начальный капитал фирмы x_0 связаны соотношением $\frac{x_0}{k} \le \frac{1}{4\alpha}, x_0 \in \left(\left(\frac{1-\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2, \left(\frac{1+\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2 \right)$.

Заключение

Как показали проведенные исследования, главный недостаток стратегии выплаты зарплаты с управлением равным 1 – низкий “потолок” прибыли. Мы видим, что линейный рост зарплаты довольно быстро ложится непомерными расходами на бюджет фирмы, замедляя ее развитие. Однако, данная стратегия никогда не приводит к полному разорению фирмы. Стратегия с управление равным 0 позволяет повысить прибыльность фирмы при выполнении указанных выше соотношений начального капитала фирмы и коэффициентов. Однако, если это соотношение не выполнено фирма либо разоряется либо уровень ее капитала падает до уровня ниже, чем при управлении равном 1.

Таким образом можем сделать вывод, что если коэффициенты a, $\frac{1}{k}$ и начальный капитал фирмы x_0 связаны соотношением $\frac{x_0}{k} \le \frac{1}{4\alpha}, x_0 \in \left(\left(\frac{1-\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2, \left(\frac{1+\sqrt{1-4\alpha\frac{x_0}{k}}}{2\alpha} \right)^2 \right)$ необходимо выбирать стратегию, когда затраты на отплату труда фирмы постоянны, во всех остальных случаях – стратегию выплаты зарплаты пропорционально капиталу фирмы.

Литература

1. Боровских А.В. Дифференциальные уравнения : учебник и практикум для академического бакалавриата : [для студ. вузов, обуч. по естественнонауч. направлениям] : в 2 ч. / А.В. Боровских, А.И. Перов. — Ч. 1. — 3-е изд., перераб. и доп. — Москва : Юрайт, 2017.— 326 с.
2. Боровских А.В. Дифференциальные уравнения : учебник и практикум для академического бакалавриата : [для студ. вузов, обуч. по естественнонауч. направлениям] : в 2 ч. / А.В. Боровских, А.И. Перов. — Ч. 2. — 3-е изд., перераб. и доп. – Москва : Юрайт, 2017. — 274 с.
3. Протасов В.Ю. Дифференциальные уравнения : курс лекций / В.Ю. Протасов. – Москва : НИУ Высшая Школа Экономики, 2018. – 79 с.
4. Андреева Е.А. Вариационное исчисление и методы оптимизации : учебное пособие / Е.А. Андреева, В.Ц. Цирулёва. – Тверь : Тверской государственный университет, 2001. – 576 с.

УДК 004.021

РЕАЛИЗАЦИЯ АЛГОРИТМА ПОСТРОЕНИЯ ДИАГРАММЫ ВОРОНОГО

О. В. Авсеева, А. И. Воротилина

Воронежский государственный университет

Введение

Диаграмма Вороного как фундаментальный инструмент вычислительной геометрии активно применяется в современной науке как в теории, так и на практике: например, в распознавании образов, робототехнике, геодезии, компьютерной графике, для численного решения задач гидро- и геомеханики.

Для таких задач традиционно используются хорошо изученные сетки на основе триангуляции расчетной области, в частности триангуляции Делоне. По сравнению с ними сетки на основе диаграммы Вороного имеют некоторые преимущества: их ячейки представляют собой выпуклые многоугольники, а использование узлов диаграммы Вороного в качестве расчетных узлов делает такую сетку в точности ортогональной.

В работе рассматривается задача построения диаграммы Вороного на плоскости.

1. Теоретические сведения о диаграмме Вороного

Диаграмма Вороного конечного множества S точек на плоскости представляет такое разбиение плоскости, при котором каждая область этого разбиения образует множество точек,

более близких к одному из элементов множества [1]. Пример диаграммы Вороного представлен на рисунке 1.

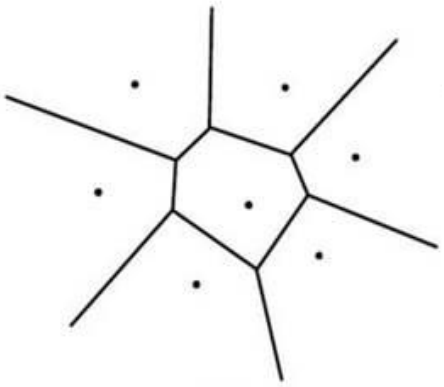


Рис. 1. Диаграмма Вороного для 7 точек

Ячейки Вороного представляют собой выпуклые многоугольники, некоторые являются бесконечными. Они также имеют название локус. Вершины многоугольников определяют вершины диаграммы Вороного, а соединяющие их отрезки – ребра диаграммы Вороного. Количество ячеек соответствует количеству заданных точек, которые имеет еще одно название — сайт.

Свойства диаграммы Вороного [1]:

Теорема 1. Каждая вершина диаграммы Вороного является точкой пересечения в точности трех ребер диаграммы.

Теорема 1 эквивалентна следующему утверждению: вершины диаграммы Вороного являются центрами окружностей, каждая из которых определяется тремя точками исходного множества, а сама диаграмма Вороного является регулярной со степенью вершин, равной трем. Определение регулярности соответствует определению в теории графов: граф является регулярным, если все вершины имеют одну и ту же степень.

Теорема 2. Для каждой вершины v диаграммы Вороного множества S окружность $C(v)$ не содержит никаких других точек множества S.

Теорема 3. Каждый ближайший сосед точки p_i в S определяет ребро в многоугольнике Вороного $V(i)$.

Теорема 4. Многоугольник $V(i)$ является неограниченным тогда и только тогда, когда точка p_i лежит на границе выпуклой оболочки множества S.

Так как лишь неограниченные многоугольники могут иметь в качестве ребер лучи, то лучи диаграммы Вороного соответствуют парам смежных точек множества S лежащих на границе выпуклой оболочки.

2. Рекурсивный алгоритм

2.1. Реализация рекурсивного алгоритма

Исходное множество точек сортируется по x координате и делится на два, примерно равных множества. Количества точек в подмножествах отличаются друг от друга не больше, чем на 1. Каждое из полученных множеств снова делится на два. Если точки имеют одинаковую координату x, точки сортируются по координате y. Так происходит до тех пор, пока в каждом из множеств останется не более двух точек. Разбиений будет не более чем $\log(n)$.

Далее, для каждого полученного множества строятся диаграммы Вороного, после чего, в порядке обратном делению, эти диаграммы объединяются в одну, как показано на рисунке 2 [2]. Полученная диаграмма Вороного и будет конечным результатом.

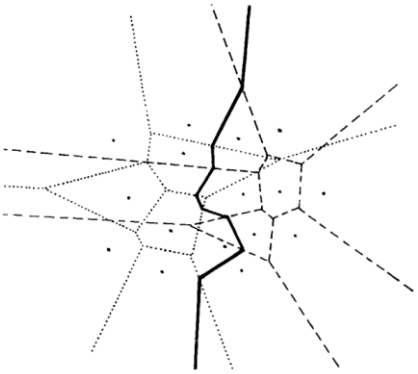


Рис. 2. Этап слияния при построении диаграммы Вороного методом “разделяй и властвуй”.

На финальном этапе спуска, построение диаграммы является построением срединного перпендикуляра между 2 точками, как представлено на рисунке 3.

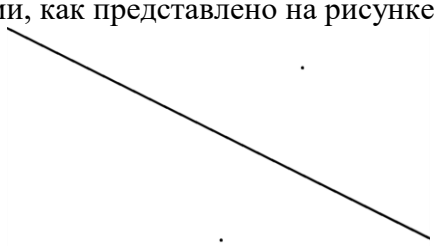


Рис. 3. Срединный перпендикуляр между 2 точками.

Объединение данных множеств и их диаграмм выполняется следующим образом:

1. Для каждого из подмножеств находится выпуклая оболочка.
2. Находятся 'верхняя и нижняя' границы данных множеств.
3. Из полученных отрезков на шаге 2, выбирается любой и обозначается за *L* (оставшийся — за *Q*). Через его середину, перпендикулярно пускается бесконечный луч. Находятся его пересечения с ячейками диаграмм Вороного исходных множеств (считается, что луч простирается вперед, то есть у него есть направление). Необходимо найти точки пересечения данного луча с соответствующими ячейками Вороного и выбрать среди них ту, что пересекается раньше. Эта точка обозначается за *M*. Конец отрезка *L*, который является центром для не пересеченной ячейки Вороного, оставляется в покое, а тот, который являлся центром ячейки, которую пересекли — обновляется: новым концом отрезка *L* станет центр ячейки Вороного, смежной по этой стороне с пересеченной ячейкой. В специальное множество *S* необходимо добавить ту часть луча, которая простирается до пересечения со стороной ячейки. Повторяется шаг 3 до тех пор, пока значения концов отрезка *L*, не станут равны значениям концов отрезка *Q*. В множестве *S* окажется непрерывный ломаный луч.
4. Получается множество S, которое представляет собой непрерывный ломаный луч. Этот луч - граница, соединяющая диаграммы Вороного двух множеств. Для получения финального результата, необходимо для диаграммы Вороного левого множества “затереть” те отрезки, которые находятся справа от полученного луча, а для диаграммы Вороного правого множества “затереть” те, которые расположены слева.

2.2. Построение выпуклой оболочки

Одним из этапов построения диаграммы Вороного является построение выпуклой оболочки множества точек.

Для построения выпуклой оболочки использовался алгоритм Грэхема-Эндрю [3].

Находится самая левая и самая правая точки A и B (если таких точек несколько, то берется самая нижняя среди левых, и самая верхняя среди правых). A и B обязательно попадут в выпуклую оболочку. Проводится через них прямая AB, разделяющая множество всех точек на верхнее и нижнее подмножества S_1 и S_2 (точки, лежащие на прямой, можно отнести к любому множеству, они всё равно не войдут в оболочку). Точки A и B относятся к обоим множествам. Построим для первого множества верхнюю оболочку, а для второго — нижнюю оболочку, объединив их, получается ответ. Чтобы получить верхнюю оболочку, необходимо отсортировать все точки по абсциссе, затем пройтись по всем точкам, рассматривая на каждом шаге кроме самой точки две предыдущие точки, вошедшие в оболочку. Если текущая тройка точек образует не правый поворот, то ближайшего соседа необходимо удалить из оболочки. Останутся только точки, входящие в выпуклую оболочку.

При поиске верхней и нижней границ общей выпуклой оболочки первого и второго подмножеств, важно, чтобы левые точки верхней и нижней границ принадлежали выпуклой оболочке первого подмножества, а правые — второй.

Проверка на правый поворот осуществляется с помощью ориентированной площади $T = x_A \times (y_B - y_C) + x_B \times (y_C - y_A) + x_C \times (y_A - y_B)$

3. Тестирование

Приведем результаты тестирования работы программы для разного количества точек на рисунках 4-6.

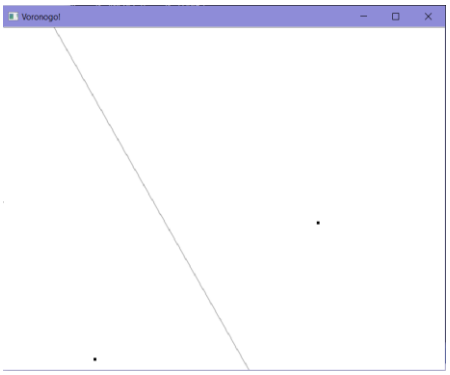


Рис. 4. Диаграмма Вороного для 2 точек

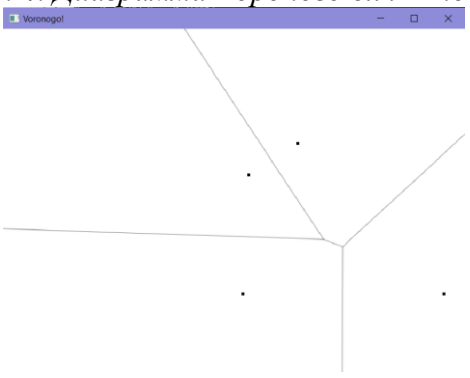


Рис. 5. Диаграмма Вороного для 4 точек

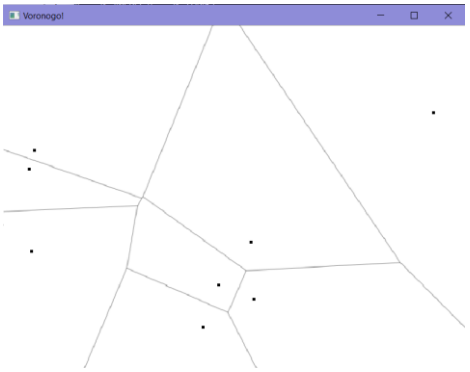


Рис. 6. Диаграмма Вороного для 8 точек

Также в таблице 1 приведены результаты тестирования времени выполнения алгоритма со случайным образом сгенерированными точками в диапазоне -1,0..1,0. Для каждого теста было сделано 10 измерений.

Таблица 1

№ теста	Количество точек	Среднее время выполнения тестов (секунды)
1	8	0.5115
2	32	0.8245
3	128	1.0073

Заключение

В результате исследования были изучены определение и свойства диаграммы Вороного, рекурсивный алгоритм ее построения. Реализован алгоритм построения диаграммы Вороного на плоскости, основанный на методе декомпозиции (“разделяй и властвуй”). Метод декомпозиции был выбран специально, так как является таким же эффективным, как и алгоритм Форчуна, однако при этом не требует при реализации сложных структур данных. Преимущество метода декомпозиции также заключается в возможности распараллеливания кода.

Литература

1. Ф. Препарата, М. Шеймос. Вычислительная геометрия: Введение. Архивная копия от 23 апреля 2011 на Wayback Machine — М.: Мир, 1989. Стр. 295
2. Shamos, M. I. Closest-point problems / M. I. Shamos, D. Hoey. — 1976 : Yale University, IEEE Sympos. Foind. Comput. Sci, 1975. — 151-162 с. — Текст : непосредственный.
3. Построение выпуклой оболочки обходом Грэхэма. — Текст : электронный // MAXimal : [сайт]. — URL: https://e-maxx.ru/algo/convex_hull_graham (дата обращения: 30.05.2022).

УДК 004.932.4

УЛУЧШЕНИЕ КАЧЕСТВА ИЗОБРАЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОБЛАЧНЫХ ТЕХНОЛОГИЙ

Е. П. Гализина, Е. В. Трофименко

Воронежский государственный университет

Введение

В последнее время все чаще и чаще можно услышать термин «облачные технологии». Несмотря на широкое распространение и частое употребление, у данного термина нет четкого и однозначного определения, так как в процессе развития облачных технологий формулировка подвергается все новым и новым изменениям и дополнениям. Под облачными технологиями понимается технология обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как Интернет-сервис.

Облачные технологии уже вошли в все сферы жизни человека: здравоохранение, образование, труд, науку и повседневную жизнь. В том числе их можно использовать в обработке изображения для улучшения его качества.

1. Облачные технологии

Суть облачных вычислений состоит в том, чтобы с их помощью получить доступ к вычислительным службам через Интернет. Облачные технологии позволяют более гибко управлять инфраструктурой. Облачные вычисления позволяют решать самые различные задачи. К достоинствам облачных вычислений относится:

- безлимитное хранилище - предоставляет неограниченные возможности хранения информации
- доступность и отказоустойчивость - доступ предоставляется всем пользователям, у которых есть интернет
- экономичность и эффективность - позволяет оплачивать только фактически потребленные ресурсы и строго по факту их использования
- простота - не требует покупки каких-либо сторонних программ
- гибкость и масштабируемость - неограниченность вычислительных ресурсов

Но, как и у любой технологии, у облачных вычислений имеются и недостатки, к которым относятся:

- постоянное соединение с сетью - вся работа осуществляется в сети
- к хранилищу данных, используя уязвимости системы, может получить доступ злоумышленник

В настоящее время существует большое количество облачных провайдеров. Из зарубежных - наиболее известны AWS(Amazon), Google Cloud Platform, из российских - больше всего на слуху Yandex Cloud, VK Cloud и SberCloud.

Рассмотрим Yandex Cloud, так как данный провайдер предоставляет пробный период пользования, имеет доступную документацию в открытом доступе и большое количество примеров использования.

В настоящее время платформа Yandex Cloud предоставляет большое количество большое количество сервисов. Для улучшения качества изображения используются Yandex Cloud Functions.

2. Улучшение изображения

В настоящее время во многих областях науки и техники проводятся работы над цифровыми изображениями. К сожалению, изображения не всегда удовлетворяют предъявляемым к ним требованиям. В таких случаях очень важную роль играет обработка полученных изображений, позволяющая довести параметры изображений до нужного уровня. Для такой обработки применяют различные методы улучшения качества изображений.

На сегодняшний день существует множество различных алгоритмов и методов, которые позволяют улучшить качество изображений. Среди методов улучшения можно выделить методы обработки в пространственной области и в частотной области. Проблема шумоподавления является одной из самых актуальных и распространенных проблем в области обработки изображений.

3. Методы шумоподавления

Шумоподавление – это процесс удаления шума, который заключается в сглаживании пикселей изображения и уменьшении мелких деталей. Целью шумоподавления является применение фильтра, который удаляет или, по крайней мере, сводит к минимуму его влияние. Для удаления шума используются следующие методы: усредненный фильтр, гауссов фильтр и

медианный фильтр.

3.1. Усредненный фильтр

В усредненном фильтре вычисляется среднее арифметическое значение в окрестности каждого рассматриваемого пикселя. Фильтр делает операцию свёртки на изображении с неким ядром, где свёртка — это вычисление нового значения пикселя, при котором учитываются значения соседних пикселей. Ядро свёртки — это квадратная матрица, где пиксель в центре этой матрицы устанавливается как среднее значение всех других пикселей, окружающих его. Чем больше размер ядра, тем более размытым будет становиться изображение, таким образом, острые края сохраняются при отбрасывании слабых.

3.2. Фильтр Гаусса

В фильтре гаусса используется нормальное распределение для вычисления преобразования, применяемого к каждому пикселю изображения. Оно похоже на предыдущее размытие, за исключением того, что вместо простого среднего мы теперь используем взвешенное среднее, где соседние пиксели, которые ближе к центральному пикселю, вносят больший «вклад» в среднее. Размытие по Гауссу является результатом размытия изображения с помощью функции Гаусса. Это широко используемый фильтр в графическом программном обеспечении.

3.3. Медианный фильтр

В медианном фильтре происходит поиск среднего значения в окрестности каждого рассматриваемого пикселя. В фильтре центральный пиксель изображения заменяется медианой всех пикселей в области ядра, в результате чего это размытие наиболее эффективно при удалении шума в стиле «соли». Этот метод нелинейной фильтрации, часто используется для удаления шума с изображения в цифровой обработке изображений, так как при определённых условиях сохраняет края при удалении шума.

4. Использование Yandex Cloud Function для методов шумоподавления

Yandex Cloud Functions – это сервис, который позволяет запускать код в виде функции в безопасном, отказоустойчивом и автоматически масштабируемом окружении без создания и обслуживания виртуальных машин. Данный сервис позволяет писать функции на нескольких языках программирования, таких как Golang, Python, Java и других. В данной работе был использован язык программирования Python для написания функций.

Для начала необходимо создать функцию в Yandex Cloud, в которой будут реализованы методы шумоподавления. Это можно сделать двумя способами. Первый - это создание функции с использованием пользовательского интерфейса Yandex Cloud. На рис.1 представлен пример создания функции первым способом.

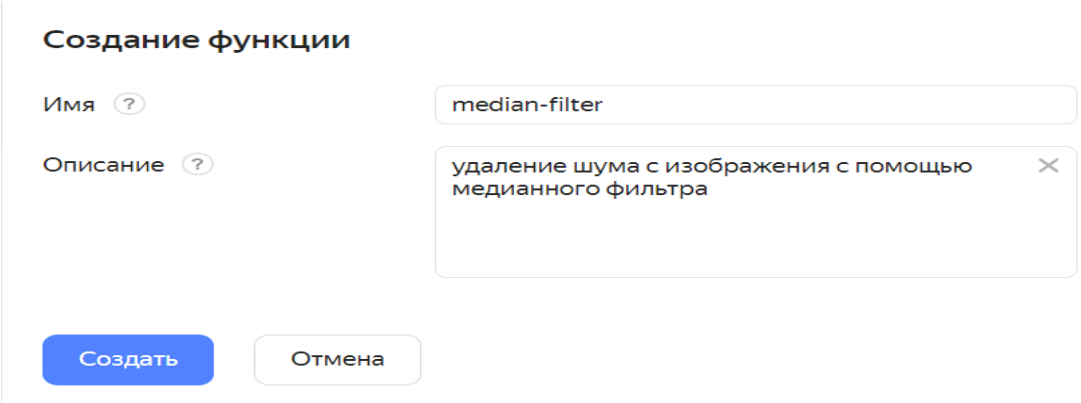


Рис.1. Создание функции с помощью пользовательского интерфейса

После нажатия кнопки “создать” функция будет создана. При каждом изменении или редактировании функции будет создаваться новая версия. На рис. 2 представлен пример успешно созданной функции.

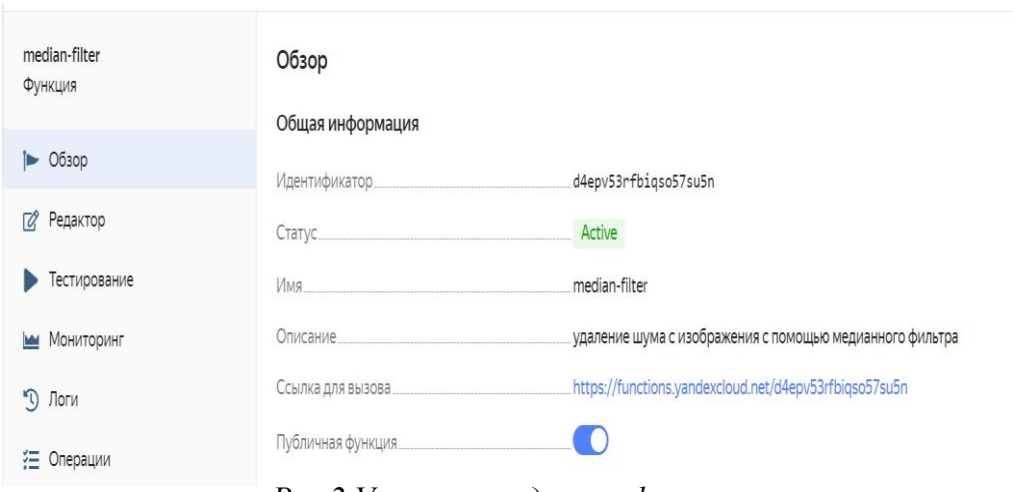


Рис.2.Успешно созданная функция

Второй способ создания функции – это использование API, которое предоставляет Yandex Cloud. На рис.3 представлен пример создания функции вторым способом.

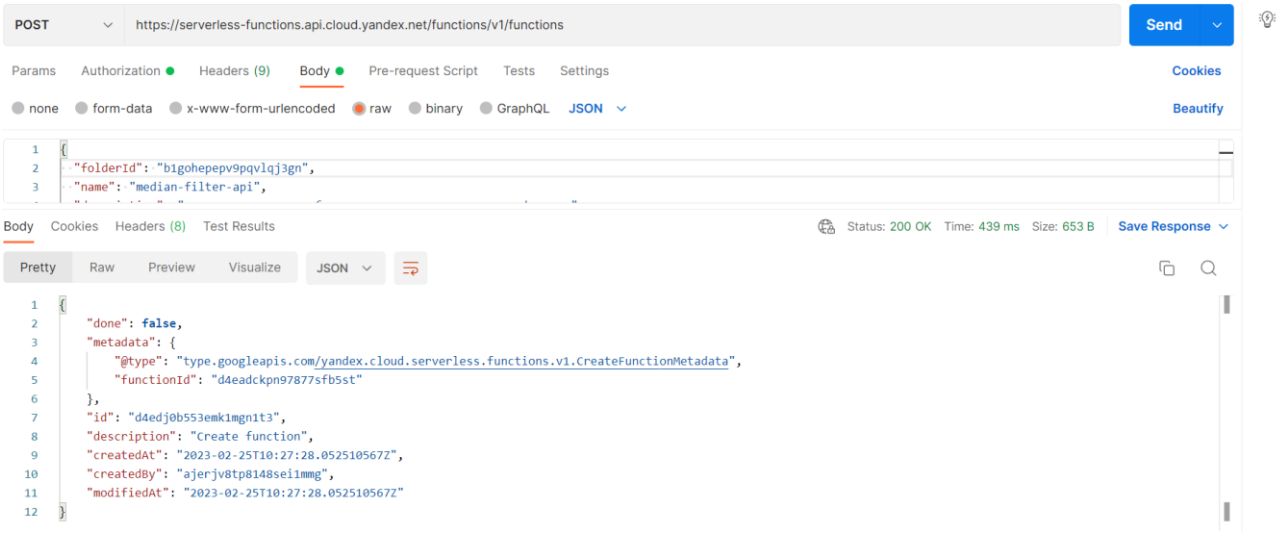


Рис.3. Создание функции с помощью API

После успешного создания функции будут отображаться в личном кабинете во вкладке функции.

Каждая функция имеет свой адрес для вызова, который будет использоваться. После того, как функции были созданы можно приступить к написанию кода для методов улучшения.

Рассмотрим создание функции для удаления шума с изображения с помощью медианного фильтра. Для написания кода необходимо перейти в нужную функцию, далее в редактор, где выбирается язык программирования, и затем в отображенное поле. Yandex Cloud Function автоматически будет устанавливать все необходимые для работы функции зависимости, которые указываются в вспомогательном файле (requirements.txt), который находится в самой функции. Такая установка зависимостей происходит при каждом создании новой версии функции. Главным при создании функции является наличие метода, который

является обработчиком запросов.

Обработчик запросов - это метод, который используется для обработки каждого вызова функции. При создании функции также необходимо указывать точку входа. Данный метод имеет следующую сигнатуру:

```
def <название_метода>(event, context):
    <тело_метода>
```

На рис. 4 представлена общая информация о функции. На данной вкладке можно увидеть полную информацию о функции: ее идентификатор, статус, имя, описание, если оно было указано при создании, ссылку для вызова и публичность функции, а также все версии функции, которые были созданы пользователем.

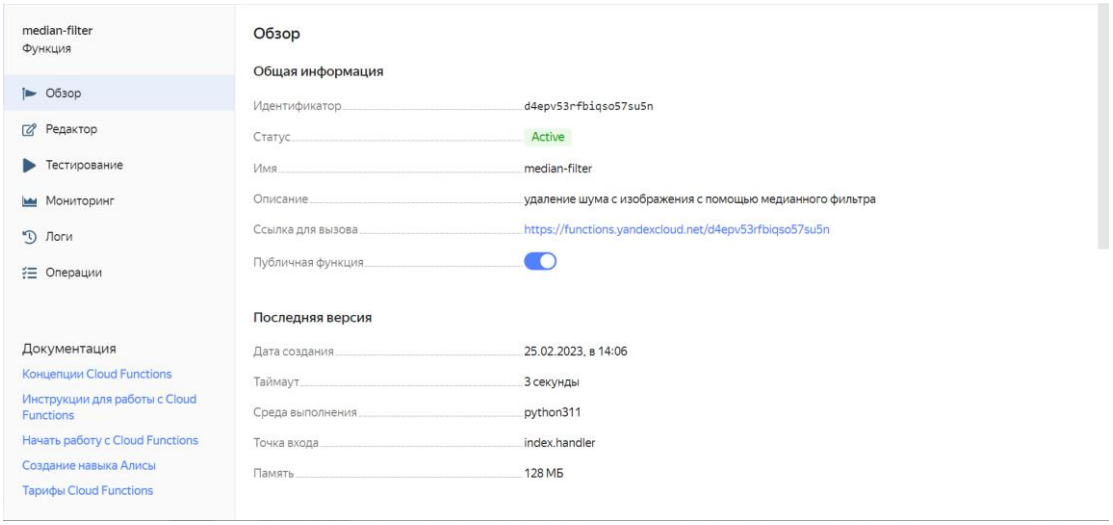


Рис.4. Общая информация о функции.

Результатом работы Yandex Cloud Function является улучшение изображение. На рис. 5а представлен пример зашумленного изображения, а на рис. 5б – улучшенное изображение при помощи медианного фильтра. Можно отметить, что после выполнения Yandex Cloud Function происходит удаление шума с изображения, получая тем самым улучшенное изображение

а) б)

Рис.5. Исходное изображение (а) и обработанное изображение (б)

По аналоги были реализованы Yandex Cloud Functions для таких методов шумоподавления как метод Гаусса и усредненный фильтр.

Одним из преимуществ использования Yandex Cloud является вкладка мониторинг для

всех Cloud Functions, которая позволяет оценить степень нагрузки на выделенные в облаке ресурсы. Yandex Cloud предоставляет информацию в виде графиков, на которых отображаются собранные метрики за выбранный промежуток времени. На вкладке мониторинг можно увидеть 6 графиков с различными метриками, а именно Requests – количество вызовов функции, Request Latency – среднее время обработки запросов, Errors – количество ошибок, возникших при выполнении функции, Inflight – количество одновременно выполняющихся вызовов функции, Used memory – используемая память при вызове функции и Initializations – частота инициализации новых экземпляров функции. На рис. 6 представлена вкладка мониторинг для написанной ранее функции.

Рис.6. Мониторинг функции

Реализованные Yandex Cloud Function удаляют шум, но сохраняют при этом детали самого изображения. Использование облачных технологий представляет собой универсальный способ удаления различных видов шума, которое за малое время способно улучшить изображение. С использованием облачных технологии время работы методов шумоподавления улучшилось в среднем 1,5 раза. Результаты сравнительного анализа быстродействия методов представлены в табл.1.

Таблица 1

Результаты сравнительного анализа

Название метода	Быстродействие алгоритма	
	Без использования облачных технологий	С использованием облачных технологий
Усредненный фильтр	428,09 мс	329,23 мс
Медианный фильтр	977,09 мс	697,85 мс
Фильтр Гаусса	707,08 мс	471,33 мс

Заключение

Были реализованы Yandex Cloud Function для улучшения качества изображения путем удаления с него шумов. Также был проведен сравнительный анализ быстродействия каждого метода.

На основе полученных результатов анализа реализованных методов шумоподавления делается вывод, что использование облачных технологий дает более качественное улучшение изображения.

Литература

1. Ярославский, Л. П. Введение в цифровую обработку изображений – Москва : Советское радио, 1979. – 315 с.

2. Документация Yandex Cloud. – Режим доступа: https://cloud.yandex.ru/docs. – (Дата обращения: 10.03.2023).
3. Технологии облачных вычислений. – Режим доступа: https://mcs.mail.ru/blog/tekhnologii-oblachnyh-vychislenij-ekonomiya-na-it?ysclid=leofqxjplz862229310 – (Дата обращения: 05.02.2023).
4. Краткий обзор облачных вычислений. – Режим доступа: https://habr.com/ru/post/111274 – (Дата обращения: 21.02.2023).

УДК 004.75

ПРИМЕНЕНИЕ БЛОКЧЕЙНА В ТРАНСПЛАНТАЦИИ ОРГАНОВ

М. А. Глинянов

Воронежский государственный университет

Введение

Технология блокчейн с каждым годом внедряется в самые разные сферы нашей жизни. Блокчейн помогает настроить процессы так, чтобы они были безопаснее, ведь данные которые в нем содержатся децентрализованы и хранятся всеми участниками сети.

Одно из возможных применений — это хранение в блокчейне и предоставление актуальной информации о донорских органах. Тысячи людей нуждаются в пересадке органов, но не все из них дожидаются своей очереди на их получение. Это происходит в частности из-за незаконных действий людей, которые хотят получить их пренебрегая очередью. Также донора может просто не найтись, так как нет единого реестра для хранения актуальной ин-формации. Эти задачи помогает решить блокчейн.

1. Разработка ПО

Целью данного исследования является разработка программного обеспечения позволяющее используя блокчейн обмениваться данными. Для этого необходимо решить следующие задачи:

- Разработка API для хранения данных в блокчейне.

- Разработка алгоритма консенсуса между участниками сети.
- Пользовательский интерфейс и API для генерации приватного и публичного ключа.

2. Принцип работы и устройства блокчейна

Блокчейн это распределенная база данных, состоящая из цепочки блоков, где каждый блок хранит значение хэш функции от предыдущего блока. Также в основе этой технологии лежат децентрализованные сети. В такой сети нет сервера, данные хранятся у каждого участника сети. Это обеспечивает надежность хранения данных. Основная цель блокчейна это возможность записывать и распространять информацию, которую нельзя редактировать и удалять [1].

Каждый блок состоит (рис. 1) из значения хэш-функции предыдущего блока, nonce, временная метка и транзакций.

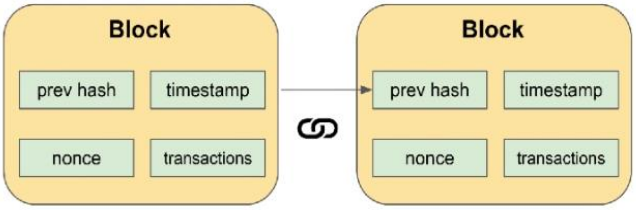


Рис. 1. Схема устройства блокчейна

Хэш-функция — позволяет преобразовать входные данные произвольного размера в строку установленной длинны. Выходная строка называется хэшем от исходных данных. Хэш позволяет однозначно идентифицировать каждый блок. Каждый раз, когда создается новый блок, в него записывается значение хэш-функции от всех данных предыдущего блока, такой подход исключает возможность изменения данных в предыдущих блоках, так как если в каком-то из блоков поменяется информация, вся последующая цепочка окажется некорректной, в свою очередь участники сети обнаружат некорректный блок и не подтвердят изменений в блокчейне.

nonce — это число, которое вычисляют участники сети, после нахождений которого происходит включения блока в блокчейн.

Транзакция — это запись в блокчейне, которая содержит в себе данные о изменении информации которую заложил в нее отправитель.

3. Консенсус и Proof-of-Work (PoW)

Правила, по которому децентрализованная сеть приходит к консенсусу (согласование между участниками сети после добавления новой информации). Консенсус достигается при генерации блока (такие транзакции считаются одобренными). Существует несколько различных подходов к консенсусу в блокчейне, два самых популярных из них это Proof-of-Work, Proof-of-Stak [2]. Каждый из них описывает свой алгоритм добавления новых блоков в основную цепочку блокчейн. Консенсус позволяет достигнуть единого мнения между участниками сети о состоянии цепочки блоков исключив мошенничества при их создании.

Proof-of-Work основывается на том, чтобы решить сложную математическую задачу, которая требует большого количества вычислительных ресурсов от участника, то есть от майнера. Необходимо найти хэш-функцию блока (рис. 2), результатом которой является некоторое значение с определенным количеством нулей в начале. После его нахождения блок

включается в основную цепочку.

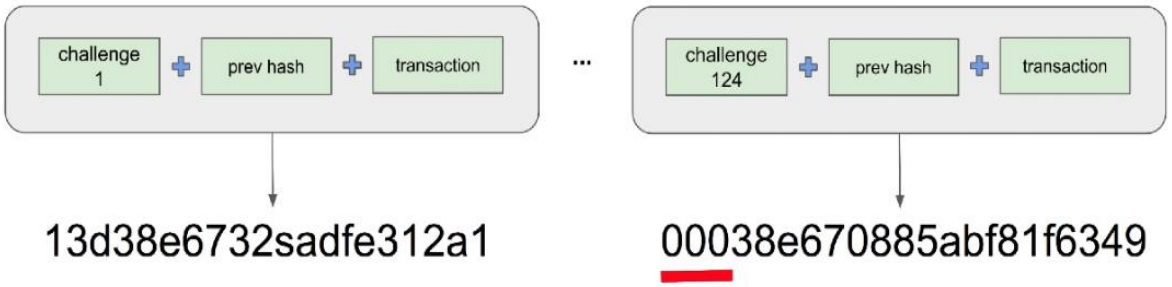


Рис. 2. Консенсус Proof-of-Work

4. Создание приватного и публичного ключа

Для того чтобы можно было отправлять какие-либо данные используя блокчейн нужен приватный и публичный ключ (рис. 3), а также адрес кошелька. Для генерации ключей используется алгоритм ECDSA базирующийся на эллиптических кривых. Приватный ключ используется для подписи транзакций, а публичный — для их проверки.

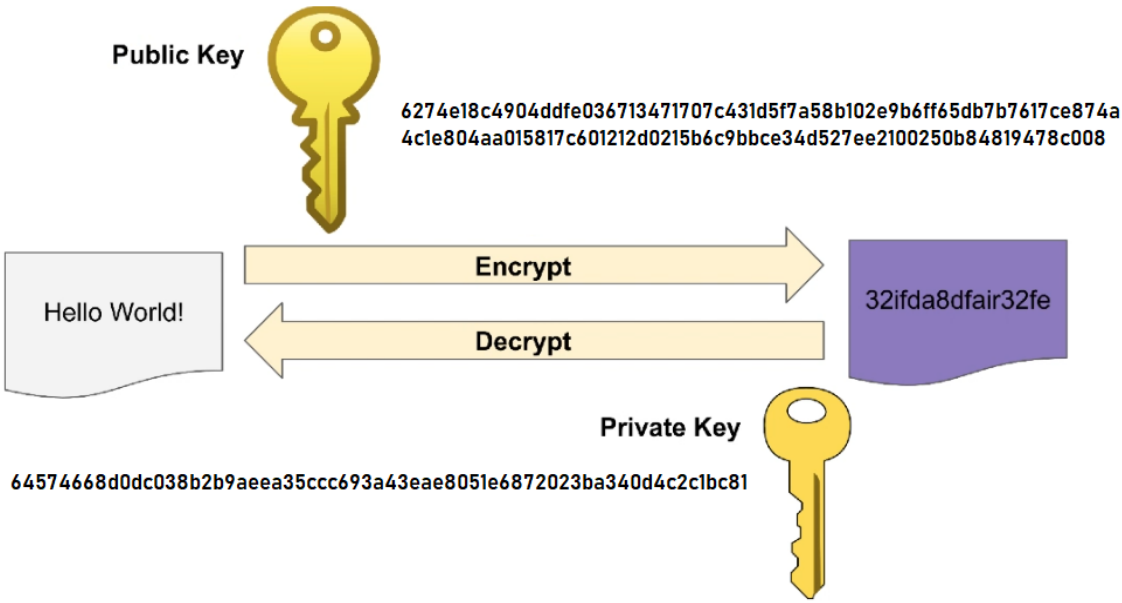


Рис. 3. Приватный и публичный ключ

Адрес генерируется с использованием ранее созданных приватного и публичного ключа, адрес однозначно ассоциируется с пользователем.
Алгоритм создания адреса по приватному и публичному ключу:

1. Выполнить хеширование SHA-256 для публичного ключа
2. Хеширование RIPEMD-160 от шага 1 (20 байт)
3. Добавить байт версии 0x00 (основная сеть)
4. Хеширование SHA-256 от 0x00 + RIPEMD-160
5. SHA-256 от шага 4
6. Взять первые 4 байта из полученного хэша (контрольная сумма)
7. Добавить контрольную сумму в конец хэша (25 байт)

8. Конвертировать полученный хэш в base58

Пример адреса: 1N8J5T1C6KxycrbDTByaFiFQ2QT8nLNGa4

5. Алгоритм проверки транзакций

До включения транзакции в блокчейн участники сети проверяют ее цифровую подпись [3].

Цифровая подпись — это сертификат подлинности, который позволяет установить, что определенная информация была создана и согласована определенным отправителем. Это технология позволяет установить целостность и неподдельность данных.

Каждый блок (рис. 4) транзакций имеет свой уникальный идентификатор, называемый хэш. Цифровая подпись в блокчейне представляет собой математический алгоритм, который используется для проверки транзакций в блокчейне.

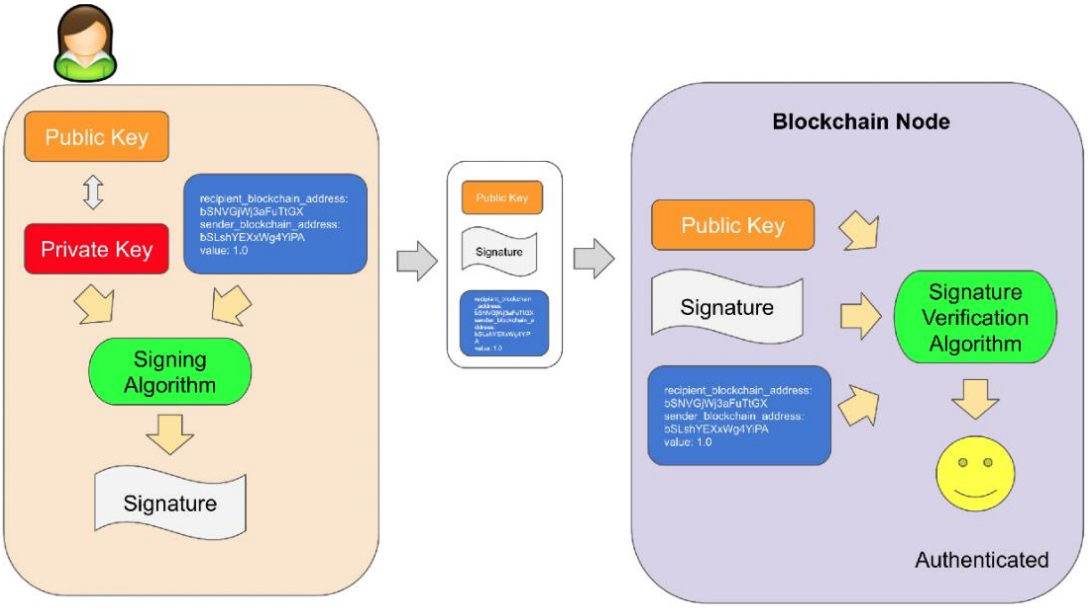


Рис. 4. Подтверждение транзакции

Полный жизненный цикл транзакции:

1. Создание транзакции и подпись транзакции приватным ключом отправителя.
2. Проверка подписи транзакции валидаторами сети. Здесь происходит сопоставление значений хэша транзакции и получившегося набора байт после проверки подписи транзакции с помощью открытого ключа. Если хэш значения совпадают это гарантирует, что транзакция создана реальным владельцем. После чего она попадает в pool транзакций, где ожидает включения в блокчейн.
3. Проверка, что у отправителя достаточно прав на владения этой информацией для совершения транзакции.
4. Подтверждённые транзакции размещаются в блоке, после чего начинается майнинг и после добычи блока (расчета значения nonce) происходит рассылка его в сеть и включение в блокчейн.

Заключение

Таким образом, было разработано ПО, с помощью которого можно создать свой

блокчейн и обмениваться информацией используя его.

Результаты исследований могут быть использовать как основа для дальнейшей разработки и применения блокчейна в узких сферах, в частности в формировании очереди на трансплантацию органов.

Литература

1. Элад Элром. Блокчейн разработчик: Руководство по проектированию, реализации, публикации и обеспечению безопасности распределенных проектов на основе блокчейна / Элад Элром Science, Business Media NY— 2019. — 360 с.
2. Arjuna Sky Kok. Hands-On Blockchain for Python Developers / Arjuna Sky Kok, Packt Publishing — 2019. — 479 с.
3. Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System / Satoshi Nakamoto — 2008. — 10

УДК 519.86

ЭКОНОМЕТРИЧЕСКАЯ МОДЕЛЬ ТРАЕКТОРИИ РАЗВИТИЯ ПРЕДПРИЯТИЯ НА ОСНОВЕ ПРОИЗВОДСТВЕННОЙ ФУНКЦИИ КОББА-ДУГЛАСА С ДОПОЛНИТЕЛЬНЫМИ ПЕРЕМЕННЫМИ

В.А. Гущина

Воронежский государственный университет

Введение

В данной статье рассматривается возможность применения производственной функции Кобба-Дугласа с дополнительными факторами интенсивного развития для моделирования процесса перехода предприятия на новую траекторию развития. В качестве фактора интенсивного развития рассматривается способность предприятия к изменению производительности труда.

1. Постановка задачи

Функция Кобба-Дугласа является одной из самых известных производственных функций, которые используются для описания производственных процессов. Данная функция отражает технологическое соотношение между объемом используемых факторов производства, а именно труда и капитала, для выпуска некоторого количества товара. Главное условие использования функции Кобба-Дугласа – стабильное и относительно устойчивое функционирование процессов предприятия. В общем виде классическая функция Кобба-Дугласа выглядит следующим образом:

$$X(t) = A \cdot K(t)^{\alpha} \cdot L(t)^{\beta}, \quad (1)$$

где $X(t)$ – выпуск предприятия в момент времени t;

$K(t)$ – объем капитала в момент времени t;

$L(t)$ – объем трудовых ресурсов в момент времени t;

A – параметр масштаба;

α – эластичность выпуска по капиталу;

β – эластичность выпуска по труду.

Эластичность факторов производства показывает, как изменение объёма каждого из них влияет на объём выпуска всего предприятия.

Параметр A отражает изменения в производственных процессах, которые позволяют предприятию увеличить количество выпускаемого товара без увеличения объёмов факторов производства.

В производственной функции Кобба-Дугласа, помимо таких факторов производства как труд и капитал, можно использовать другие факторы, которые могут оказывать немалое влияние на объём выпуска продукции. В данной работе предлагается учитывать в производственной функции такой фактор интенсивного развития, как способность предприятия к изменению производительности труда $G(t)$. С учетом данного фактора модель будет выглядеть следующим образом:

$$X(t) = A \cdot K(t)^{\alpha} \cdot L(t)^{\beta} \cdot G(t-1)^{\gamma}, \quad (2)$$

где $G(t-1) = P(t-1) / P(t-2)$;

γ – эластичность выпуска по способности предприятия к изменению производительности труда;

$P(t)$ – производительность труда в момент времени t.

Производительность труда определяется количеством произведенной продукции на одного рабочего за единицу времени.

Для построения модели производственной функции Кобба-Дугласа с дополнительной переменной необходимо разработать алгоритм оценки неизвестных параметров A, α, β, γ.

2. Алгоритм оценки неизвестных параметров производственной функции Кобба-Дугласа с дополнительной переменной

Поиск неизвестных параметров A, α, β, γ можно осуществить с помощью метода регрессионного анализа на базе статистических данных за ретроспективный период деятельности предприятия.

Для начала производится логарифмирование уравнения производственной функции:

$$\ln X(t) = \ln A + \alpha \ln K(t) + \beta \ln L(t) + \gamma \ln G(t-1) \quad (3)$$

и введение замены переменных:

– $A = e^{\theta}$;

– $x_1 = \ln K(t)$;

– $x_2 = \ln L(t)$;

– $x_3 = \ln G(t)$.

В результате замены получается линейная модель:

$$Y(t) = \theta + \alpha x_1 + \beta x_2 + \gamma x_3. \quad (4)$$

В процессе построения уравнения регрессии проводится вычисление оценок неизвестных параметров A, α, β, γ, проверка гипотез о значимости построенного уравнения, проверка гипотез о значимости коэффициентов регрессии и анализ ошибок для построенного уравнения. Если проведенный анализ показывает адекватность полученного уравнения регрессии, то его можно использовать для выработки требований к значениям интенсивного фактора при формировании определенной траектории результирующего показателя.

Вместе с основным уравнением регрессии строится уравнение регрессии, которое описано следующей формулой:

$$L(t+1) = B \cdot L(t)^{\delta} \cdot G(t-1)^{\mu}. \quad (5)$$

Для оценки коэффициентов данной модели также осуществляется переход к линейной модели и применение процедуры линейного регрессионного анализа.

Предположим, что значения результирующего показателя в период времени $[t_0, t_1]$ изменялись за счет интенсивного фактора по определенной траектории:

$$X(t) = X_n + k(t - t_n)^p. \quad (6)$$

Интервал $[t_n, t_1]$ можно разделить на несколько частей и для границ каждой из частей указать значения $X(t)$, соответствующее рассматриваемой траектории, как показано в таблице 1:

Таблица 1

Разбиение периода выхода на нужную траекторию на интервалы

t_{n+1}	t_{n+2}	t_{n+3}	…	t_{n+m}
$X(t_{n+1})$	$X(t_{n+2})$	$X(t_{n+3})$	…	$X(t_{n+m})$

Дальнейшие расчеты можно провести в соответствии со следующими формулами:

$$X(t_{n+1}) = A \cdot K(t_n)^{\alpha} \cdot L(t_n)^{\beta} \cdot G(t_n)^{\gamma}, \quad (7)$$

$$G(t_n) = \left(\frac{X(t_{n+1})}{A \cdot K(t_n)^{\alpha} \cdot L(t_n)^{\beta}}\right)^{\frac{1}{\gamma}}, \quad (8)$$

$$L(t_{k+1}) = L(t_k)^{\delta} \cdot G(t_{k-1})^{\mu}, \quad (9)$$

$$X(t_{n+2}) = A \cdot K(t_n)^{\alpha} \cdot L(t_{n+1})^{\beta} \cdot G(t_n)^{\gamma}, \quad (10)$$

…

$$X(t_{n+m}) = A \cdot K(t_n)^{\alpha} \cdot L(t_{n+m-1})^{\beta} \cdot G(t_{n+m-2})^{\gamma}, \quad (11)$$

$$G(t_{n+m-2}) = \left(\frac{X(t_{n+m})}{A \cdot K(t_n)^{\alpha} \cdot L(t_{n+m-1})^{\beta}}\right)^{\frac{1}{\gamma}}. \quad (12)$$

Заключение

Таким образом, учёт такого фактора, как способность предприятия к изменению производительности труда в производственной функции Кобба-Дугласа позволит оценить возможность предприятия выйти на новую траекторию развития, то есть оптимизировать использование производственных факторов, чтобы достичь максимальных объёмов производства при минимальных затратах.

Литература

1. Азарнова Т.В. Математические модели и методы оценки и управления ресурсной устойчивостью развития предприятия / Т.В. Азарнова, Н.Г. Аснина, Л.А. Щепин // Системы управления и информационные технологии. – 2021. – № 3(85). – С. 54–59.

2. Артюхов В.В. Общая теория систем: самоорганизация, устойчивость, разнообразие, кризисы. / В.В. Артюхов – 3-е изд. Москва: Книжный дом «Либроком», 2012.

3. Бекренев И.В. Методические аспекты формирования адаптивного механизма устойчивого развития предприятия на основе целевого комплексного подхода / И.В. Бекренев, Я.Н. Лозовская // Вестник РУДН. Серия: Экономика. – 2017. – № 2. – С. 233–241.

4. Кутовая А.С. Анализ подходов к определению понятия «устойчивое развитие предприятия» / А.С. Кутовая // Вестник Саратовского государственного социально-экономического университета. – 2012. – № 5 (44). – С. 39–43.

УДК 004.42

АНАЛИЗ ТЕХНОЛОГИЙ MONGODB, EXPRESS, REACT JS, NODE JS В РАЗРАБОТКЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ УЧЕТА НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ ПРЕПОДАВАТЕЛЯ

В. М. Деревянкина

Воронежский государственный университет

Введение

В современном мире использование информационных технологий в образовании и научных исследованиях становится все более распространенным. При этом существует множество различных технологий, которые могут быть использованы в разработке информационных систем для учета научно-исследовательской деятельности преподавателя. В данном контексте особенно интересны технологии MongoDB, Express, React JS и Node JS, которые образуют современный стек MEAN-технологий и широко применяются в веб-разработке.

1. Этапы разработки

Разработка информационной системы для учета научно-исследовательской деятельности преподавателя может быть разбита на несколько этапов.

1. Планирование проекта. На этом этапе определяются требования к будущей системе, ее функциональность, цели и задачи. Важно учесть все потребности пользователей, а также технические требования к разработке.
2. Проектирование системы. На этом этапе определяется архитектура системы, ее компоненты и взаимодействие между ними.
3. Реализация системы. Создание кода, используя выбранные технологии и следуя плану разработки.
4. Тестирование системы. На этом этапе система проверяется на соответствие требованиям и качество ее работы.

2. Анализ front-end фреймворка React JS

2.1. Краткое описание и этапы разработки

React JS — это открытый и свободный фронт-энд фреймворк, который позволяет разрабатывать пользовательские интерфейсы для веб-приложений. React был разработан Facebook и используется в таких известных компаниях, как Netflix, Airbnb, Instagram и другие.

Этапы разработки front-end:

1. Сбор требований и планирование: этот этап включает определение целей проекта, анализ требований пользователей и выбор технологий для разработки.
2. Проектирование интерфейса и прототипирование: на этом этапе создаются макеты и прототипы интерфейса, которые будут использоваться для дальнейшей разработки.
3. Разработка и тестирование: этот этап включает написание кода и тестирование его работоспособности на разных устройствах и браузерах.
4. Оптимизация: на этом этапе оптимизируется производительность и скорость работы приложения, чтобы улучшить пользовательский опыт.

2.2. Преимущества

1. Высокая производительность: React использует виртуальный DOM (Document Object Model), это концепция программирования, в которой «виртуальное» представление пользовательского интерфейса хранится в памяти и синхронизируется с «настоящим» DOM при помощи библиотеки ReactDOM, также react использует механизмы оптимизации рендеринга, что обеспечивает быстрое отображение большого объема данных.
2. Переиспользование компонентов: компонентный подход в React позволяет создавать гибкие и многократно используемые компоненты, что значительно упрощает процесс разработки и ускоряет его.
3. Легкая интеграция: React может использоваться в сочетании с другими фреймворками и библиотеками, что обеспечивает гибкость и расширяемость.
4. Наличие большого сообщества: React имеет широкую поддержку со стороны сообщества разработчиков, что обеспечивает доступ к большому количеству инструментов, документации и решений.

2.3. Недостатки

1. Необходимость использования дополнительных инструментов: React требует использования дополнительных инструментов, таких как Babel и Webpack, для сборки и оптимизации проекта.
2. Сложность масштабирования: при работе с крупными проектами может возникнуть сложность масштабирования и управления состоянием приложения.
3. Сложность интеграции с бэк-эндом: React является только фронт-энд фреймворком и требует интеграции с бэк-эндом, что может быть сложным при работе с определенными технологиями.

Преимущества React, такие как эффективное управление состоянием и масштабируемость, позволяют создавать гибкий и производительный пользовательский интерфейс. Однако надо учитывать недостатки React, такие как высокий порог вхождения и опасность переусложнения кода, чтобы избежать возможных проблем в будущем.

3. Анализ back-end фреймворка Express, работающий внутри среды исполнения Node JS

3.1. Краткое описание и этапы разработки

Express является одним из самых популярных back-end фреймворков для Node.js. Он обеспечивает простоту и гибкость в разработке веб-приложений, включая возможность создания API.

Этапы разработки back-end:

1. Сбор требований и планирование: этот этап включает определение целей проекта, анализ требований пользователей и выбор технологий для разработки.
2. Проектирование базы данных: на этом этапе проектируется структура базы данных, которая будет использоваться для хранения данных приложения.
3. Разработка и тестирование серверного кода: этот этап включает написание серверного кода и тестирование его работоспособности.
4. Интеграция с фронтендом: на этом этапе серверный код интегрируется с кодом front-end для создания полноценного приложения.
5. Оптимизация: на этом этапе оптимизируется производительность и скорость работы серверного кода, чтобы улучшить работу приложения и обеспечить масштабируемость.

3.2. Преимущества

1. Простота и гибкость: Express имеет минимальный набор функций и позволяет разработчикам создавать свои собственные решения, что делает его очень гибким и удобным для использования.
2. Широкая поддержка: Express является одним из самых популярных Node.js фреймворков и имеет большое сообщество, что обеспечивает доступ к большому количеству инструментов, библиотек и решений.
3. Легкая настройка и установка: Express имеет простую установку и настройку, что делает его быстрым в использовании.
4. Middleware: Express обеспечивает механизм middleware, который позволяет легко добавлять дополнительную функциональность в приложение, такую как обработка ошибок, аутентификация и авторизация.

3.3. Недостатки

1. Отсутствие жесткой структуры: Express не имеет строгой структуры проекта, что может привести к сложностям при работе с крупными проектами.
2. Необязательное следование принципу единственной ответственности: в Express нет принудительного следования принципу единственной ответственности, что может привести к возникновению проблем в разработке.
3. Сложность масштабирования: при работе с крупными проектами может возникнуть сложность масштабирования и управления состоянием приложения.
4. Отсутствие стандартизации: в отличие от некоторых других фреймворков, таких как Ruby on Rails, Express не имеет стандартов и лучших практик, что может усложнить разработку веб-приложений.

Преимущества Express, такие как его простота, гибкость и большое сообщество, позволяют быстро создавать серверные приложения и настраивать их в соответствии с конкретными потребностями проекта. Однако стоит учитывать недостатки Express, такие как недостаток стандартов и недостаток защиты по умолчанию, чтобы избежать возможных проблем в будущем. Учитывая это, можно использовать Express эффективно и создать качественный проект.

4. Реализация проекта «Информационная система для учета научно-исследовательской деятельности преподавателя»

Вышеописанные преимущества и недостатки данных фреймворков были учтены при реализации проекта «Информационная система для учета научно-исследовательской деятельности преподавателя». Его суть заключается в создание комплексной системы для учета научно-исследовательской деятельности преподавателей. Она включает в себя следующие функциональные компоненты: регистрация пользователя, загрузка научно-исследовательских работ, а также их просмотр. Такая система позволит преподавателям эффективно организовать свою научную деятельность, повысить эффективность работы и получить больше результатов в своих исследованиях.

Заключение

Результатом исследования стал анализ средств front-end и back-end разработки web-приложений. В ходе исследования были выявлены особенности вышеописанных средств разработки. На основе этого анализа выбраны необходимые средства для реализации клиент-серверного web-приложения.

Литература

1. JavaScript-библиотека для создания пользовательских интерфейсов // reactjs.org URL: https://ru.reactjs.org/ (дата обращения: 27.02.2023).
2. Скринкаст по Node.js // javascript.ru URL: https://learn.javascript.ru/screencast/nodejs (дата обращения: 12.03.2023).
3. Начало работы с Express // METANIT.COM URL: https://metanit.com/web/nodejs/4.1.php (дата обращения: 12.03.20

УДК 004.94

ГЕНЕРАЦИЯ УРОВНЕЙ-ЛАБИРИНТОВ В UNITY3D

В. Ю. Евсеенко, Е.В. Трофименко

Воронежский государственный университет

Введение

Разработка игр проходит в несколько этапов, одним из которых является создание уровней. Поскольку создавать все уровни вручную достаточно трудозатратно, а в некоторых проектах может и не требоваться вовсе, разработчиками часто применяются различные алгоритмы для автоматизации данного процесса. Поскольку ландшафт генерируется с нуля, то такую автоматизацию можно применять для создания большого количества уровней, при этом не увеличивая размер самой игры.

2. Обзор основных видов алгоритмов генерации лабиринтов

1.1 Алгоритмы, работающие с графами

Лабиринт может быть сгенерирован с помощью заданного заранее расположения клеток (чаще всего это прямоугольная сетка, но возможны и другие варианты) со стенами между ними. Эта сетка может считаться связным графом с гранями, которые обозначают возможные места для стен и узлами, обозначающими клетки [1].

Если подграф не связный, то это значит, что эти места графа потрачены впустую, поскольку они никак не влияют на построение алгоритма. Если же в графе содержатся циклы, то могут существовать несколько путей для некоторых узлов. Из-за этого генерация лабиринтов часто представляет собой построение случайного остовного дерева. Циклы, которые могут сбить с толку наивных решателей лабиринтов, могут быть введены путем добавления случайных ребер к результату в ходе работы алгоритма. На рисунке 1 показан результат работы алгоритма Прима для создания лабиринта.

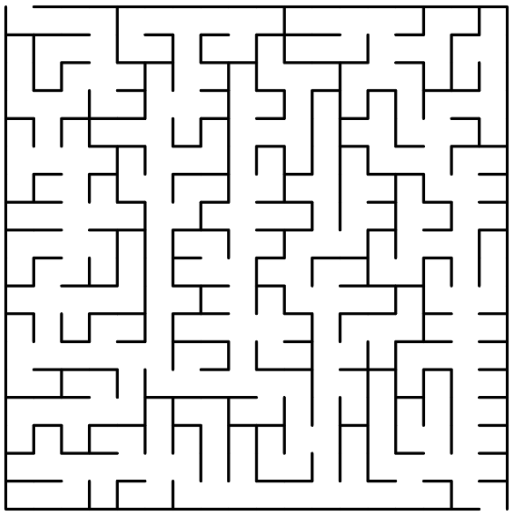


Рис.1. Результат работы алгоритма Прима для создания лабиринта

1.2 Метод рекурсивного деления

Лабиринты можно создавать с помощью рекурсивного деления, алгоритм которого работает следующим образом: Алгоритм начинается с пустого пространства лабиринта без стен, называемое комнатой. Эта комната разделяется с помощью случайно расположенной стены(или нескольких стен), где каждая стена содержит так же случайно расположенный проход внутри него. Далее этот процесс рекурсивно продолжается до тех пор, пока все подкомнаты не будут минимального размера. Этот метод образует лабиринты с длинными прямыми стенами, из-за которых становится видно те места, которых следует избегать при прохождении лабиринта.

Например, в прямоугольном лабиринте строятся две стены, перпендикулярные друг другу. Эти стены разделяют пространство лабиринта на четыре более маленьких комнаты, разделённых стеной. Случайно выбираются три стены из четырёх и добавляется проход шириной в одну клетку в каждой из них. Это рекурсивно продолжается до тех пор, пока каждая комната будет шириной в одну клетку в любом из двух направлений. На рисунке 2 показан пример данного алгоритма.

Например, в прямоугольном лабиринте постройте в случайных точках две стены, перпендикулярные друг другу. Эти две стены делят большую камеру на четыре меньшие камеры, разделенные четырьмя стенами. Выберите три из четырех стен наугад и откройте отверстие шириной в одну ячейку в случайной точке каждой из трех. Продолжайте таким образом рекурсивно, пока каждая камера не будет иметь ширину в одну ячейку в любом из двух направлений.

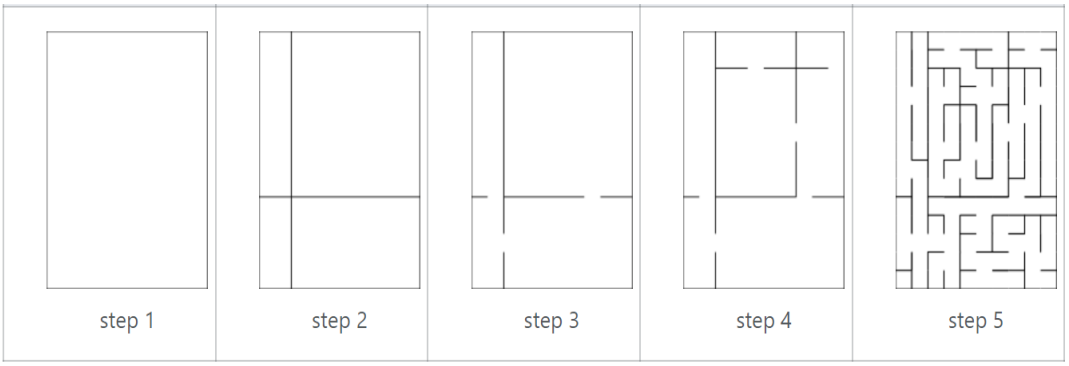


Рис. 2. Пример применения рекурсивного деления

1.3 Алгоритм поиска в глубину

В данной статье рассматривается реализация алгоритма поиска в глубину с бэктрекингом для создания лабиринтов. Этот алгоритм часто применяется со стэком, такой подход является одним из самых простых в реализации способов генерации лабиринта.

Алгоритм начинает работу с большой сетки (по сути большой шахматной доски), каждая клетка которой создаётся с четырьмя стенками. Начиная со случайной клетки, алгоритм выбирает случайную соседнюю клетку, которая ещё не была посещена[2]. Стена между двумя клетками убирается, а новая клетка помечается посещённой и добавляется в стек, чтобы в дальнейшем иметь возможность к ней вернуться. При этом клетка у которой нет соседей будет считаться тупиком. Попадая в тупик алгоритм возвращается по пути, до тех пор, пока не посетит клетку с непосещённым соседом. Путь продолжится с помощью этой клетки, при этом создаётся новое ответвление. Выполнение этих операций будет выполняется до тех пор, пока каждая клетка не будет посещена, что приведёт нас к начальной клетке.

Алгоритм, описанный выше включается в себя глубокую рекурсию, которая может вызвать переполнение стэка на некоторых компьютерах. Данный метод может быть переделан в цикл благодаря сохранению информации для бэктрекинга в самом лабиринте. Это также позволить быстро показать решение, начиная в любой точке и возвращаясь к началу. Пример лабиринта, созданного данным алгоритмом представлен на рисунке 3.

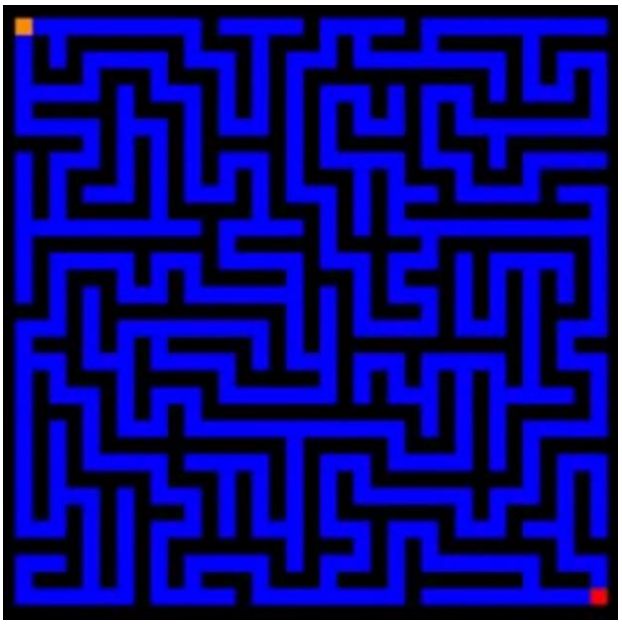


Рис.3. Пример генерации с помощью алгоритма поиска в глубину

Недостаток рекурсивного подхода – это глубина рекурсии, в худшем случае алгоритм будет перепроходить по каждой клетке заданной сетки, что может превысить максимальную глубину стэка рекурсии в некоторых случаях.

В качестве решения данной проблемы тот же метод бэктрекинга может быть выполнен со специальным явным стэком, который может увеличиваться без превышения лимита глубины. Этот метод имеет следующую последовательность действий:

1. Выбирается начальная клетка, она помечается посещённой и попадает в стэк
2. Пока стэк не пуст:
    a. клетка убирается из стэка и она становится текущей
    b. если у текущей клетки есть соседи, которые не были посещены:
        I. клетка попадает в стек;
        II. выбирается одна из непосещённых клеток-соседей;
        III. удаляется стена между текущей клеткой и выбранной;
        IV. выбранная клетка помечается как посещённая и помещается в стэк.

3. Реализация алгоритма

Для данной работы был реализован алгоритм поиска в глубину в Unity3D, с помощью языка программирования C#. Для реализации алгоритма были созданы несколько заготовок комнат (стартовая, конечная, комнаты с врагами или сундуками) и класс DungeonGenerator, отвечающий за создание лабиринта, а также класс RoomBehaviour, который определяет стены и двери в комнатах.

При запуске приложения сначала создаётся начальная комната, далее случайным образом выбираются варианты следующих комнат, а при завершении создания лабиринта создаётся последняя комната(конечная). Результаты работы алгоритма представлены на рисунке 4.

Рис.4. Пример генерации с помощью алгоритма в Unity3D

На данном рисунке виден готовый вид созданного лабиринта. Начало лабиринта в правом верхнем углу, а конец - слева внизу.

Рис.5. скриншот готовой программы

Заключение

Были рассмотрены два вида алгоритмов: алгоритмы, работающие с помощью графов и метод рекурсивного деления, применяемых для генерации лабиринтов и примеры их

реализации. Также алгоритм поиска в глубину с бэктрекингом был реализован в Unity3D для наглядной демонстрации работы данного алгоритма.

Литература

1. Wilson, David Bruce "Generating random spanning trees more quickly than the cover time"/ Wilson, David Bruce: Philadelphia: ACM, 1996. – p.296-303.
2. Maze generation algorithm- depth first search. – Режим доступа: https://www.algosome.com/articles/maze-generation-depth-first.html – (Дата обращения 29.03.2023)
3. Maze Generation Algorithms. – Режим доступа:: https://visualize-it.github.io/maze_generation/simulation.html– (Дата обращения 29.03.2023)

УДК 519.61

ВЫЧИСЛЕНИЕ ОБРАТНОГО К РАЗНОСТНОМУ ОПЕРАТОРУ

Л. И. Еникеева

Воронежский государственный университет

Введение

Разностные уравнения и операторы изучаются давно и имеют многочисленные приложения [1, 2, 5]. Настоящая статья посвящена разностному оператору $\left(Ax\right)_n = \sum_{k=-q}^{p} a_k x_{n-k}$ с матричными коэффициентами a_k размера $d \times d$. Оператор A рассматривается в пространстве $l_p\left(\mathbb{Z}, \mathbb{C}^d\right)$. Обратимость этого оператора равносильна однозначной разрешимости задачи об ограниченных решениях. Условия обратимости оператора A хорошо известны [7, теорема 4.5.7]. Построение обратного оператора сводится к представлению характеристической функции $\psi(z) = \left[\sum_{k=-q}^{p} a_k z^k\right]^{-1}$ в виде степенного ряда $\psi(z) = \sum_{k=-\infty}^{+\infty} b_k z^k$, $|z| = 1$. Обсуждается численный алгоритм решения этой задачи, состоящий в вычислении не более, чем $(p+q)d$ вычетов матричнозначной функции ψ. Поскольку нахождение вычетов является некорректной задачей [6], точность ответа получается ниже точности, с которой проводятся вычисления. Предлагаются способы повышения точности.

1. Характеристическая функция обратного оператора

Оператор A, действующий в $l_p = l_p\left(\mathbb{Z}, \mathbb{C}^d\right)$, $1 \le p \le \infty$, по правилу

$$\left(Ax\right)_n = \sum_{k=-q}^{p} a_k x_{n-k},$$

где $a_k \in \mathbb{C}^{d\times d}$, $k = -q, \ldots, p$, ограничен [7, предложение 1.6.8, следствие 4.4.10]. Ему соответствует характеристическая функция или пучок

$$\varphi(z) = a_{-q}\frac{1}{z^q} + a_{-q+1}\frac{1}{z^{q-1}} + \ldots + a_{-1}\frac{1}{z} + a_0 + a_1 z + \ldots + a_{p-1}z^{p-1} + a_p z^p.$$

Предполагается, что оператор обратим в пространстве $l_p\left(\mathbb{Z}, \mathbb{C}^d\right)$, $1 \le p \le \infty$. Известно [7, теорема 5.2.6], что обратный оператор A^{-1} действует в $l_p = l_p\left(\mathbb{Z}, \mathbb{C}^d\right)$, $1 \le p \le \infty$, по аналогичному правилу:

$$\left(A^{-1}x\right)_n = \left(Bx\right)_n = \sum_{k=-\infty}^{\infty} b_k x_{n-k},$$

где $b_k \in \mathbb{C}^{d\times d}$, $\sum_{k=-\infty}^{\infty} \|b_k\| < +\infty$. Для характеристической функции ψ обратного оператора A^{-1} верно соотношение

$$\psi(z) = \left[\varphi(z)\right]^{-1}, \ |z| = 1.$$

Различные варианты следующего предложения хорошо известны, см., например, [4, лемма 12.2].

Предложение 1. *Множество собственных значений пучка*

$$\varphi_{\text{inv}}(\lambda) = \lambda^p I + \lambda^{p-1} a_1 + \ldots + \lambda a_{p-1} + a_p$$

совпадает с множеством собственных значений матрицы

$$\mathbf{A} = \begin{pmatrix} 0 & I & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & I \\ -a_p & -a_{p-1} & \cdots & -a_2 & -a_1 \end{pmatrix}.$$

В дальнейшем всегда будем предполагать, что матрица a_p обратима.

Следствие 2. *Пусть матрица a_{-q} обратима. Тогда собственные значения пучка*

$$\tilde{\varphi}(z) = a_{-q} + a_{-q+1} z + \ldots + a_{-1} z^{q-1} + a_0 z^q + a_1 z^{q+1} + \ldots + a_{p-1} z^{p+q-1} + a_p z^{p+q}$$

совпадают с числами $z = 1/\lambda$, где λ — ненулевые собственные значения пучка

$$\varphi_{\text{inv}}(\lambda) = \lambda^{p+q} + \lambda^{p+q-1} b_1 + \ldots + \lambda^{p+1} b_{q-1} + \lambda^p b_q + \lambda^{p-1} b_{q+1} + \ldots + \lambda b_{p+q-1} + b_{p+q},$$

где $b_i = a_{-q}^{-1} a_{-q+i}$, $i = 1, \ldots, p+q$.

Доказательство. Представим функцию $\tilde{\varphi}$ в виде

$$\tilde{\varphi}(z) = \sum_{i=-q}^{p} a_i z^{i+q} = a_{-q} \left(\sum_{i=-q+1}^{p} a_{-q}^{-1} a_i z^{i+q} + I \right) = a_{-q} \left(\sum_{i=-q+1}^{p} b_{q+i} z^{i+q} + I \right).$$

Для $z \neq 0$ и $\lambda \neq 0$ утверждение очевидно. Число $z = 0$ не является собственным значением. Но $\lambda = 0$ может быть собственным значением при условии, что матрица b_{p+q} необратима, или, что то же самое, необратима матрица a_p. По этой причине нулевые собственные значения λ мы исключаем из рассмотрения.

Следствие 3. *Собственные значения пучка*

$$\tilde{\varphi}(z) = a_{-q} + a_{-q+1} z + \ldots + a_{-1} z^{q-1} + a_0 z^q + a_1 z^{q+1} + \ldots + a_{p-1} z^{p+q-1} + a_p z^{p+q}$$

совпадают с собственными значениями пучка

$$\varphi_{\text{inv}}(\lambda) = \lambda^{p+q} + \lambda^{p+q-1} b_1 + \ldots + \lambda^{p+1} b_{p-1} + \lambda^p b_p + \lambda^{p-1} b_{p+1} + \ldots + \lambda b_{p+q-1} + b_{p+q},$$

где $b_i = a_{-p}^{-1} a_{p-i}$, $i = 1, \ldots, p+q$.

Доказательство. Проведём следующие преобразования:

$$\tilde{\varphi}(z) = \sum_{i=-q}^{p} a_i z^{i+q} = a_p \left(\sum_{i=-q}^{p-1} a_p^{-1} a_i z^{i+q} + I z^{p+q} \right) = a_p \left(\sum_{i=-q}^{p-1} b_{p-i} z^{i+q} + I z^{p+q} \right),$$

где $b_i = a_{-p}^{-1} a_{p-i}$, $i = 1, \ldots, p+q$. Следовательно,

$$\det\left(\tilde{\varphi}(z)\right) = \det\left(a_p\right) \det\left(\sum_{i=-q}^{p-1} b_{p-i} z^{i+q} + I z^{p+q} \right).$$

Отсюда получаем, что $\det\left(\tilde{\varphi}(z)\right) = 0$ тогда и только тогда, когда равен нулю определитель

$$\det\left(\sum_{i=-q}^{p-1} b_{p-i} z^{i+q} + I z^{p+q} \right) = 0.$$

Возможны два случая: матрица b_{p+q} может быть либо обратимой, либо необратимой, что соответственно равносильно обратимости или необратимости матрицы a_{-q}. Последнее означает наличие собственного значения $z = 0$.

Предложение 4. *Если все собственные значения пучка* $\tilde{\varphi}(z) = \sum_{i=-q}^{p} a_i z^{i+q}$ *являются простыми, то функцию* $\tilde{\psi}(z) = \left[\tilde{\varphi}(z)\right]^{-1}$ *можно представить в виде*

$$\tilde{\psi}(z) = \sum_{j=0}^{h-l} f_j z^j + \sum_{k=1}^{l} \frac{c_k}{z - z_k},$$

где f_j, c_k — *некоторые матрицы размера* $d \times d$, z_k — *собственные значения пучка* $\tilde{\varphi}$, $l \le (p+q)d$, $h \le (p+q)d - (p+q)$ (*если* $h < l$, *то первая сумма отсутствует*).

2. Алгоритм вычислений

Рассмотрим оператор

$$\left(\tilde{A}x\right)_n = a_{-q} y_n + a_{-q+1} y_{n-1} + \ldots + a_{-1} y_{n-q+1} + a_0 y_{n-q} + a_1 y_{n-q-1} + \ldots + a_{p-1} y_{n-p-q+1} + a_p y_{n-p-q},$$

получающийся заменой $x_{n-i} = y_{n-i-q}$, $i = -q, \ldots, p$, в операторе A. Очевидно, что операторы A и $\tilde{A}$ обратимы одновременно. Характеристическая функция оператора $\tilde{A}$ имеет следующий вид:

$$\tilde{\varphi}(z) = a_{-q} + a_{-q+1} z + \ldots + a_{-1} z^{q-1} + a_0 z^q + a_1 z^{q+1} + \ldots + a_{p-1} z^{p+q-1} + a_p z^{p+q}.$$

Следующая диаграмма коммутативна в силу коммутативности её верхнего и нижнего квадратов:

$$\begin{array}{ccc} A & \xrightarrow{x_{n-i}=y_{n-i-q}} & \tilde{A} \\ \downarrow & & \downarrow \\ \phi & \xrightarrow{\times z^q} & \tilde{\phi} \\ \downarrow & & \downarrow \\ \psi & \xleftarrow{\times z^q} & \tilde{\psi} \end{array}$$

Действительно, нетрудно видеть, что пучки операторов A и $\tilde{A}$ связаны соотношением

$$\tilde{\varphi}(z) = z^q I \varphi(z) = \varphi(z) z^q I = z^q \varphi(z).$$

Тогда

$$\tilde{\psi}(z) = \left[\tilde{\varphi}(z)\right]^{-1} = \left[z^q I \varphi(z)\right]^{-1} = \left[\varphi(z)\right]^{-1} \left[z^q I\right]^{-1} = \left[\varphi(z)\right]^{-1} \frac{1}{z^q} I = \frac{1}{z^q} \left[\varphi(z)\right]^{-1},$$

$$\psi(z) = \left[\varphi(z)\right]^{-1} = z^q \frac{1}{z^q} \left[\varphi(z)\right]^{-1} = z^q \tilde{\psi}(z).$$

Задача состоит в нахождении явной формулы, определяющей обратный оператор A^{-1}. Для того чтобы получить представление для обратного оператора A^{-1}, будем искать формулу, определяющую обратный оператор $\tilde{B} = \tilde{A}^{-1}$, отвечающий характеристической функции $\tilde{\psi}(z) = \left[\tilde{\varphi}(z)\right]^{-1}$. Для этого приведем $\tilde{\psi}$ к виду $\tilde{\psi}(z) = \sum_{k=-\infty}^{+\infty} \tilde{b}_k z^k$, где $\tilde{b}_k$ — некоторые матрицы размера $d \times d$, а затем найдем ψ по формуле:

$$\psi(z) = z^q \tilde{\psi}(z) = \sum_{k=-\infty}^{+\infty} b_k z^k.$$

Воспользуемся предложением 4 и произведем следующие преобразования. Разложим $\tilde{\psi}(z)$ в степенной ряд на единичной окружности:

$$\tilde{\psi}(z) = \sum_{i=0}^{h-l} f_i z^i + \sum_{k=1}^{l} \frac{c_k}{z - z_k} = \sum_{i=0}^{h-l} f_i z^i + \sum_{k=1}^{l} c_k \begin{cases} \sum_{j=0}^{+\infty} \frac{z_k^j}{z^{j+1}} \text{ при } |z_k| < 1, \\ \sum_{j=0}^{+\infty} -\frac{z^j}{z_k^{j+1}} \text{ при } |z_k| > 1 \end{cases} =$$

$$= \sum_{i=0}^{h-l} f_i z^i + \sum_{k=1}^{l} c_k \begin{cases} \sum_{j=-\infty}^{-1} \frac{z_k^{-j-1}}{z^{-j}} \text{ при } |z_k| < 1, \\ -\sum_{j=0}^{+\infty} \frac{z^j}{z_k^{j+1}} \text{ при } |z_k| > 1 \end{cases} =$$

$$= \sum_{i=0}^{h-l} f_i z^i + \sum_{k=1}^{l} c_k \begin{cases} \sum_{j=-\infty}^{-1} z_k^{-j-1} z^j \text{ при } |z_k| < 1, \\ -\sum_{j=0}^{+\infty} z_k^{-j-1} z^j \text{ при } |z_k| > 1. \end{cases}$$

На единичной окружности ряды, стоящие после фигурных скобок, абсолютно сходятся; это позволяет внести c_k под знак внутренней суммы и поменять порядок суммирования:

$$\tilde{\psi}(z) = \sum_{i=0}^{h-l} f_i z^i + \begin{cases} \sum_{j=-\infty}^{-1} \sum_{k=1}^{l} c_k z_k^{-j-1} z^j \text{ при } |z_k| < 1, \\ -\sum_{j=0}^{+\infty} \sum_{k=1}^{l} c_k z_k^{-j-1} z^j \text{ при } |z_k| > 1. \end{cases}$$

Отсюда получаем формулу для характеристической функции ψ оператора, обратного к оператору A,

$$\psi(z) = z^q \tilde{\psi}(z) = \sum_{i=q}^{h-l+q} f_i z^i + \begin{cases} \sum_{j=-\infty}^{-1+q} \sum_{k=1}^{l} c_k z_k^{-j-1} z^j \text{ при } |z_k| < 1, \\ -\sum_{j=q}^{+\infty} \sum_{k=1}^{l} c_k z_k^{-j-1} z^j \text{ при } |z_k| > 1. \end{cases}$$

Находим коэффициент b_j, стоящий перед z^j в формуле для ψ:

$$b_j = g_j + \begin{cases} \sum_{k=1}^{l} c_k z_k^{-j-1} \text{ при } |z_k| < 1,\ j < q, \\ -\sum_{k=1}^{l} c_k z_k^{-j-1} \text{ при } |z_k| > 1,\ j \geq q, \end{cases} \tag{1}$$

где

$$g_j = \begin{cases} f_j, \text{ если } j \leq h - l + q \text{ и } j \geq q, \\ 0, \text{ если } j > h - l + q \text{ или } j < q. \end{cases}$$

Итак, оператор A^{-1} имеет вид

$$\left(A^{-1}x\right)_n = \sum_{k=-\infty}^{+\infty} b_k x_{n-k}.$$

Выражение для b_j из формулы (1) содержит неизвестные матрицы c_k. В нашей предыдущей работе коэффициенты c_k вычислялись по формуле, представляющей собой правило нахождения вычетов в полюсе первого порядка: $c_k = \lim_{z \to z_k} \psi(z)(z - z_k)$, в которой по причине некорректности задачи вычисления такого предела в смысле А. Н. Тихонова [6]

предел заменялся приближенным значением $c_k \approx \left[\varphi\left(z_k+\varepsilon\right)\right]^{-1}\varepsilon,$ где $\varepsilon>0$ — маленькое число. Основной новой идеей настоящей работы является наблюдение: более точное приближение можно получить, используя одну из формул (2) или (3):

$$c_k \approx \left(\left[\varphi\left(z_k+\varepsilon\right)\right]^{-1}-\left[\varphi\left(z_k-\varepsilon\right)\right]^{-1}\right)\frac{\varepsilon}{2}, \tag{2}$$

$$c_k \approx \frac{1}{3}\left(\left[\varphi\left(z_k+\varepsilon u\right)\right]^{-1}\varepsilon u+\left[\varphi\left(z_k+\varepsilon u^2\right)\right]^{-1}\varepsilon u^2+\left[\varphi\left(z_k+\varepsilon u^3\right)\right]^{-1}\varepsilon u^3\right), \tag{3}$$

где $\varepsilon>0$ — маленькое число, $u=e^{i\frac{2\pi}{3}}$ удовлетворяет уравнению $u^3=1.$

Действительно, известно, что значение вычета функции в полюсе первого порядка совпадает с коэффициентом C_{-1} разложения этой функции в ряд Лорана в окрестности полюса. Поэтому имеем

$$\left[\varphi\left(z_k+\varepsilon\right)\right]^{-1}\varepsilon-C_{-1}=\psi\left(z_k+\varepsilon\right)\varepsilon-C_{-1}=\left(\frac{C_{-1}}{\varepsilon}+C_0+C_1\varepsilon+C_2\varepsilon^2+\ldots+C_i\varepsilon^i+\ldots\right)\varepsilon-C_{-1}=$$

$$=C_0\varepsilon+C_1\varepsilon^2+C_2\varepsilon^3+\ldots+C_i\varepsilon^{i+1}+\ldots=C_0\varepsilon+o\left(\varepsilon\right),$$

$$\left(\left[\varphi\left(z_k+\varepsilon\right)\right]^{-1}-\left[\varphi\left(z_k-\varepsilon\right)\right]^{-1}\right)\frac{\varepsilon}{2}-C_{-1}=\left(\left(\frac{C_{-1}}{\varepsilon}+C_0+C_1\varepsilon+C_2\varepsilon^2+\ldots+C_i\varepsilon^i+\ldots\right)-\right.$$

$$\left.-\left(\frac{-C_{-1}}{\varepsilon}+C_0-C_1\varepsilon+C_2\varepsilon+\ldots+\left(-1\right)^i C_i\varepsilon^i+\ldots\right)\right)\frac{\varepsilon}{2}-C_{-1}=$$

$$=\left(\frac{2C_{-1}}{\varepsilon}+2C_1\varepsilon+2C_3\varepsilon^3+\ldots\right)\frac{\varepsilon}{2}-C_{-1}=C_1\varepsilon^2+o\left(\varepsilon^2\right).$$

Таким образом, видим, что формула (2) дает более точное приближение, чем формула, по которой вычисления производились ранее. Далее рассмотрим формулу (3):

$$\frac{1}{3}\left(\left[\varphi\left(z_k+\varepsilon u\right)\right]^{-1}\varepsilon u+\left[\varphi\left(z_k+\varepsilon u^2\right)\right]^{-1}\varepsilon u^2+\left[\varphi\left(z_k+\varepsilon u^3\right)\right]^{-1}\varepsilon u^3\right)-C_{-1}=$$

$$=\frac{1}{3}\left(\left(\frac{C_{-1}}{\varepsilon u}+C_0+C_1\varepsilon u+C_2\varepsilon^2u^2+\ldots+C_j\varepsilon^ju^j+\ldots\right)\varepsilon u+\right.$$

$$+\left(\frac{C_{-1}}{\varepsilon u^2}+C_0+C_1\varepsilon u^2+C_2\varepsilon^2u^4+\ldots+C_j\varepsilon^ju^{2j}+\ldots\right)\varepsilon u^2+$$

$$\left.+\left(\frac{C_{-1}}{\varepsilon u^3}+C_0+C_1\varepsilon u^3+C_2\varepsilon^2u^6+\ldots+C_j\varepsilon^3u^{3j}+\ldots\right)\varepsilon u^3\right)-C_{-1}=$$

$$=\frac{1}{3}\left(3C_{-1}+\left(\varepsilon u+\varepsilon u^2+\varepsilon u^3\right)C_0+\left(\varepsilon^2u^2+\varepsilon^2u^4+\varepsilon^2u^6\right)C_1+\left(\varepsilon^3u^3+\varepsilon^3u^6+\varepsilon^3u^9\right)C_2+\right.$$

$$\left.+\ldots+\left(\varepsilon^{j+1}u^{j+1}+\varepsilon^{j+1}u^{2j+2}+\varepsilon^{j+1}u^{3j+3}\right)C_i+\ldots\right)-C_{-1}.$$

Последнее с учетом равенства

$$u^{j+1}+u^{2j+2}+u^{3j+3}=\begin{cases}3, \text{ если } (j+1) \text{ делится на 3},\\ 0, \text{ если } (j+1) \text{ не делится на 3}\end{cases}$$

равно

$$C_{-1}+\varepsilon^3C_2+\varepsilon^6C_5+\ldots+\varepsilon^{3r}C_{3r-1}+\ldots-C_{-1}=C_2\varepsilon^3+o\left(\varepsilon^3\right),$$

где $r=0,1,2,\ldots.$

Можно заметить, что формула (3) дает более точное приближение, чем формула (2). Но в то же время вычисления по формуле (3) требуют обращения большего числа матриц.

3. Численные эксперименты

В проводившихся численных экспериментах брались матрицы a_i, $i = -q, \ldots, p$, со случайными элементами, распределенными равномерно в интервале $(-1;1)$. Рассматривался оператор $\tilde{A}$. Собственные значения z_k пучка $\tilde{\varphi}$ вычислялись в соответствии с предложением 1 и следствием 2. Наконец, по формуле (1) вычислялись коэффициенты b_j оператора A^{-1}, обратного к оператору A. Проводилась проверка путем умножения полученного обратного оператора A^{-1} на исходный оператор A в соответствии с предложением 1.

Бралось по несколько наборов матриц a_i, $i = -q, \ldots, p$, размеров 5×5, 10×10, 20×20 и 50×50. Для каждого набора a_i, $i = -q, \ldots, p$, проводились вычисления с ε, равными 10^{-5}, 10^{-6}, 10^{-7} и 10^{-8} с использованием формул (2) и (3) для вычисления c_k.(Для формулы (3) дополнительно бралось ε, равное 10^{-4}.) Вычислялось несколько коэффициентов e_k оператора

Таблица 1

Результаты для $(Ax)_n = Ix_n + a_1x_{n-1} + a_2x_{n-2} + a_3x_{n-3}$

d	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
5	$3.3\cdot10^{-9}$	$2.3\cdot10^{-10}$	$2.9\cdot10^{-9}$	$3.1\cdot10^{-8}$
10	$1.6\cdot10^{-8}$	$2.3\cdot10^{-9}$	$1.7\cdot10^{-8}$	$1.7\cdot10^{-7}$
20	$1.6\cdot10^{-7}$	$9.8\cdot10^{-9}$	$9\cdot10^{-8}$	$6.2\cdot10^{-7}$
50	$2.5\cdot10^{-6}$	$5.1\cdot10^{-8}$	$4.2\cdot10^{-7}$	$3.8\cdot10^{-6}$

$E = AB$. В качестве *точности* результата бралось наибольшее из чисел $\|e_0 - I\|$, $\|e_k\|$, $k = -10, \ldots, 10$. Результаты численных экспериментов для формулы (2) для операторов $(Ax)_n = Ix_n + a_1x_{n-1} + a_2x_{n-2} + a_3x_{n-3}$ и $(Ax)_n = Ix_{n+1} + a_0x_n + a_1x_{n-1} + a_2x_{n-2}$, где I — единичная матрица размера $d\times d$, приведены в табл. 1 и табл. 2 соответственно, для формулы (3) – в табл. 3 и табл. 4 соответственно.

Заключение

Таким образом, в настоящей работе предложен способ вычисления вычетов матричнозначных функций, который применен для нахождения обратного к разностному оператору. Эффективность метода подтверждается численными экспериментами.

Вычисления проводились в пакете «Математика» [3, 8].

Настоящая работа написана под руководством Курбатова Виталия Геннадьевича, профессора кафедры САиУ ВГУ.

Таблица 2

Результаты для $(Ax)_n = Ix_{n+1} + a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2}$

d	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
5	$1 \cdot 10^{-8}$	$4 \cdot 10^{-10}$	$3.4 \cdot 10^{-9}$	$4.5 \cdot 10^{-8}$
10	$4.1 \cdot 10^{-8}$	$1.5 \cdot 10^{-9}$	$3 \cdot 10^{-8}$	$2.1 \cdot 10^{-7}$
20	$1.4 \cdot 10^{-7}$	$3.6 \cdot 10^{-9}$	$4.2 \cdot 10^{-8}$	$4.5 \cdot 10^{-7}$
50	$4.4 \cdot 10^{-6}$	$6 \cdot 10^{-8}$	$2.6 \cdot 10^{-7}$	$2.8 \cdot 10^{-6}$

Таблица 3

Результаты для $(Ax)_n = Ix_n + a_1 x_{n-1} + a_2 x_{n-2} + a_3 x_{n-3}$

d	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
5	$1.5 \cdot 10^{-9}$	$2.1 \cdot 10^{-11}$	$2.4 \cdot 10^{-10}$	$4.8 \cdot 10^{-9}$	$3.4 \cdot 10^{-8}$
10	$8.4 \cdot 10^{-9}$	$1.4 \cdot 10^{-10}$	$5.7 \cdot 10^{-10}$	$1.2 \cdot 10^{-8}$	$8.3 \cdot 10^{-8}$
20	$1.3 \cdot 10^{-7}$	$4.3 \cdot 10^{-10}$	$4.6 \cdot 10^{-9}$	$3.4 \cdot 10^{-8}$	$5.4 \cdot 10^{-7}$
50	$3.2 \cdot 10^{-6}$	$3.7 \cdot 10^{-9}$	$1.9 \cdot 10^{-8}$	$1.7 \cdot 10^{-7}$	$1.3 \cdot 10^{-6}$

Таблица 4

Результаты для $(Ax)_n = Ix_{n+1} + a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2}$

d	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$
5	$5.3 \cdot 10^{-10}$	$5.3 \cdot 10^{-11}$	$3.5 \cdot 10^{-10}$	$4.4 \cdot 10^{-9}$	$4.9 \cdot 10^{-8}$
10	$5.9 \cdot 10^{-9}$	$2.4 \cdot 10^{-10}$	$1.2 \cdot 10^{-9}$	$2.7 \cdot 10^{-8}$	$2.4 \cdot 10^{-7}$
20	$2 \cdot 10^{-7}$	$1 \cdot 10^{-9}$	$1 \cdot 10^{-8}$	$1 \cdot 10^{-7}$	$1 \cdot 10^{-6}$
50	$2 \cdot 10^{-6}$	$5.6 \cdot 10^{-9}$	$2 \cdot 10^{-8}$	$3.1 \cdot 10^{-7}$	$5 \cdot 10^{-6}$

Литература

1. Баскаков, А. Г. Исследование линейных дифференциальных уравнений методами спектральной теории разностных операторов и линейных отношений / А. Г. Баскаков // Успехи математических наук. — 2013. — Т. 68, № 1 (409). — С. 77—128.

2. Курант, Р. О разностных уравнениях математической физики / Р. Курант, К. О. Фридрихс, Г. Леви // Успехи математических наук. — 1941. — № 8. — С. 125— 160.

3. Курбатов, В. Г. Пакет “Математика” в прикладных научных исследованиях : учебное пособие / В. Г. Курбатов, В. Е. Чернов. — Воронеж : Издательский дом ВГУ, 2016. — 240 с.

4. Маркус, А. С. Введение в спектральную теорию полиномиальных операторных пучков / А. С. Маркус. — Кишинев : Штиинца, 1986. — 260 с.

5. Романко, В. К. Разностные уравнения : учебное пособие / В. К. Романко. — 4 изд. — М. : Лаборатория знаний, 2020. — 115 с.

6. Тихонов, А. Н. Методы решения некорректных задач / А. Н. Тихонов, В. Я. Арсенин. — М. : Наука, 2022. — 288 с.

7. Kurbatov, V. G. Functional differential operators and equations / V. G. Kurbatov. — Dordrecht : Kluwer Academic Publishers, 1999. — Vol. 473 of Mathematics and its Applications. — xx+433 p.

8. Wolfram, S. The Mathematica book / S. Wolfram. — Fifth edition. — New York : Wolfram Media, 2003. — 1488 p.

УДК 004.92

ОБЗОР АЛГОРИТМОВ ВОССТАНОВЛЕНИЯ ИЗОБРАЖЕНИЙ, ИСКАЖЕННЫХ ПЕРСПЕКТИВНЫМ ПРЕОБРАЗОВАНИЕМ

В.А Енокян, Д. Е. Черняев, А.В. Шатковская, П.В. Черныховский

Воронежский государственный университет

Аннотация: В данной статье представлен обзор различных подходов к восстановлению изображений, искаженных перспективным преобразованием и представлены выводы для анализа правильного подхода для решения конкретных проблем.

Ключевые слова: восстановление изображений, искажение изображения.

Введение

Перспективное искажение возникает из-за перспективной проекции 3D-сцены на 2D-поверхность. Исправление искажения изображения без потери нужной информации является одной из основных задач в области компьютерного зрения. Это происходит из-за того, что объект наблюдения расположен под некоторым углом к видеокамере или другому аналогичному устройству. Перспектива — это проекция реального видения. Для многих применений перспективы потребуется параллельная проекция, потому что параллельная проекция упрощает извлечение и применение закономерностей изображений. Однако они (параллельные проекции) не параллельны в перспективной проекции, а следовательно их определение и применение имеют большое значение. Это ведет к устранению искажения перспективы.[2]

Перспективное искажение возникает из-за перспективной проекции 3D-сцены на 2D-поверхность. Рассмотрена задача оценки искажения перспективы по одиночному кадру изображения неструктурированной среды и задача перспективной коррекции, которая является как количественно точной, так и визуально приятной. В статье представлено несколько методов коррекции перспективного искажения и целью этой статьи является обзор результативности данных методик и алгоритмов.

1. Причины искажения изображений

В процессе доступа к изображению, поскольку изображение на саму систему влияют такие вещи, как нелинейность, угол съемки и другие факторы, при которых произойдет геометрическое искажение изображения. Геометрические искажения подразделяются на системное искажение и несистемное искажение. Искажение системы является регулярным и может быть предсказано, а несистемные искажения носят случайный характер. При подготовке количественного анализа изображения, точная геометрическая коррекция будет сначала реализована в искаженном изображении чтобы не повлиять на точность количественного анализа. Искажение — это одна из системных аберраций изображения и идеальная система визуализации, которая является не только четкой в визуализации, но и также удовлетворяет соотношению подобия изображения объекта. В случае, когда система способна обеспечить четкое изображение, она описывает отношение отображения между идеальной точкой изображения и фактическим положением точки изображения, а именно,

$$(x, y, z) = f(x_{d,} y_{d,} z_{d}) \quad (1)$$

и суть коррекции искажения заключается в определении такого отношения отображения.

Геометрическое искажение изображений, собранных системой машинного зрения, является нелинейной, поэтому такие нелинейные преобразования могут быть представлены полиномиальным преобразованием между координатами:

$$x_d = \sum_{i=0}^{n} \sum_{j=0}^{n-1} a_{ij} x^i y^j \quad (2)$$

$$y_d = \sum_{i=0}^{n} \sum_{j=0}^{n-1} b_{ij} x^i y^j \quad (3)$$

Точность коррекции связана с n, и чем больше n то, тем выше точность коррекции, но при увеличении n, величина расчета коррекции будет резко увеличена.

2. Метод, основанный на плоской гомографии

На перспективном изображении объекты с одинаковым размером кажутся больше, когда они ближе к точке обзора, и меньше, когда они дальше. Следовательно, количество пикселей для представления более удаленных объектов на изображении требует меньшего количества пикселей по сравнению с количество пикселей, необходимое для представления более близких объектов на изображении. Пример: когда объекты наклонены к зрителю, он уходит дальше от точки зрения. Следовательно, меньшее число пикселей означает меньше количество информации для обработки изображения. Перспективные изображения приводят ко многим неоднозначным результатам, когда мы склонны измерять размеры объектов на картинке. Прямоугольная плоскость $ABCD$ на рисунке 1(а) представлена как $A_1B_1C_1D_1$ в случае перспективного искажения. Так формируются изображения предметов в процессе получения изображения без искажения.

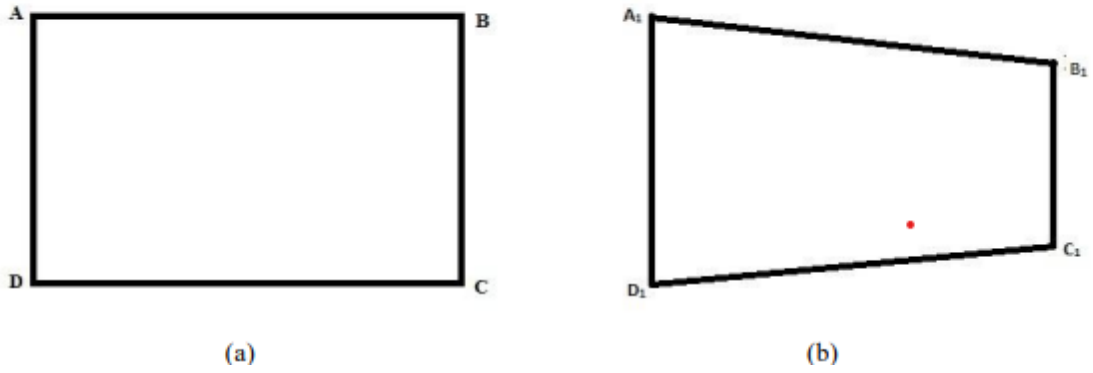


Рис.1. Изображение без/с перспективного искажения

Преобразование перспективы превращает перспективную проекцию в параллельную проекцию. Перспективное преобразование используется для замены объема вида (объема в форме призмы) на прямоугольную форму. Параллельная проекция преобразованных примитивов такая же, как и непреобразованное изображение в перспективной проекции.

Чтобы построить параллельную проекцию изображения из перспективно искаженного изображения, в плоскости используется гомография [6]. Гомографию можно вычислить, зная относительные положения четырех точек на искаженном в перспективе изображении и их позиции в преобразованном изображении. Четыре угловые точки необходимы для реализации перспективного преобразования. Угловой метод [2] обнаруживает все доступные угловые точки. Оценка, которая рассчитывается для каждого пикселя при помощи углового детектора основана на двух собственных значениях матрицы. Выражение для его вычисления не является

произвольным, а основывается на наблюдении за тем, как выражение меняется с разными собственными значениями. Среди доступных угловых точек необходимо выбрать четыре желаемые угловые точки. Плоскую гомографию интересуют только четыре точки, т. е. точка пересечения ребер. Это получается путем взятия минимального и максимального значений в массиве угловых точек.

Угловые точки далее сортируются по часовой стрелке. Четыре угловые точки $C_1(X1,Y1), C_2(X2,Y2), C_3(X3,Y3), C_4(X4,Y4)$ прямоугольного объекта перспективного изображения используются в методе плоской гомографии, как на рисунке 2 (a). Полученное исправленное изображение имеет углы $V_1(x1,y1), V_2(x2,y2), V_3(x3,y3), V_4(x4,y4)$, где $(x1,y1)=(0,0),(x2,y2)=(L,1),(x3,y3)=(L,B),(x4,y4)=(1,B)$ как на рисунке 2(b).

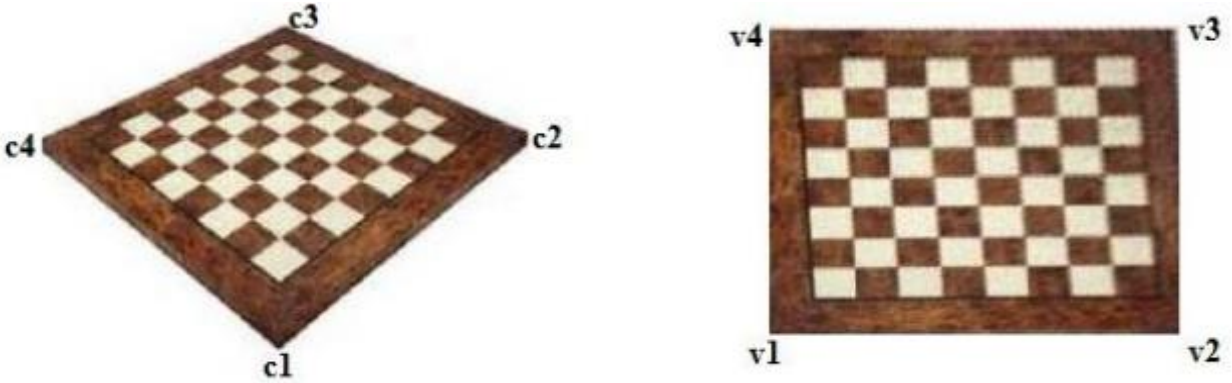


Рис. 2. Исправление перспективы плоской гомографией

Плоская гомография широко используется для отображения границы перспективы на границу прямоугольника. что является необходимым шагом к перспективному преобразованию. 2D-гомография определена однородной матрицей 3x3, которая отображает любую точку $p(x,y)$ на плоскости π в соответствующую ей точку $p'(x',y')$ на π' как:

$$p' = H \bullet p \tag{4}$$

Гомография описывается гомографическим преобразованием, когда p и p' преобразовываются в однородные координаты.

$$\begin{pmatrix} wx' \\ wy' \\ 1 \end{pmatrix} = \begin{pmatrix} * & * & * \\ * & * & * \\ * & * & * \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{5}$$

Вычисяем (4), чтобы применить гомографию H. Гомографию можно применить для исправления перспективного искажение изображения путем нахождения гомографии H по заданным парам p и p'. Вывод можно получить, найдя сначала соответствие для одной угловой точки, а затем расширив его до четырех точек.

Рассмотрим соответствие для одной точки $p(x,y) \rightarrow p'(x',y')$.

Уравнение (5) принимает следующий вид:

$$\begin{pmatrix} wx' \\ wy' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (6)$$

Вычисляя x' и y':

$$x' = \frac{ax+by+c}{gx+hy+i}, y' = \frac{dx+ey+f}{gx+hy+i} \quad (7)$$

$$\begin{aligned} ax+by+c-gxx'-hyy'-ix' &= 0 \\ dx+ey+f-gxx'-hyy'-ix' &= 0 \end{aligned} \quad (8)$$

Составляем соответствующие уравнения для других 3 точек.

Запишем уравнения выше как матрицу A_i. $h=0$ для $i=1$, 2 где $h=\begin{bmatrix} a & b & c & d & e & f & g & h & i \end{bmatrix}^T$ является вектором с неизвестными коэффициентами из H и A_i это матрица 2x9 составленная из известных координат $xi, yi, x'i, y'i$:

$$\begin{bmatrix} xi & yi & 1 & 0 & 0 & 0 & -x'ixi & -x'yi & -x'i \\ 0 & 0 & 0 & xi & yi & 1 & -y'iyi & -y'yi & -y'i \end{bmatrix} \quad (9)$$

Результирующее уравнение можно представить в виде:

$$p' = H \bullet p \rightarrow A_i \bullet h = 0 \text{ для } i = 1,2,3,4. \quad (10)$$

3. Метод, основанный на билинейной интерполяции

Пусть $[f(x,y)]$ - исходное изображение, а $[f(u,v)]$ - искаженное изображение в прямоугольной системе координат. Перспективное искажение изображения – это геометрическое искажение изображения пространственного типа и выражаем это искажение через квадратичный полином:

$$u = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 \quad (11)$$

$$v = b_0 + b_1x + b_2y + b_3x^2 + b_4xy + b_5y^2 \quad (12)$$

Также после оценки параметра (a_i, b_i) можно получить функцию искажения, и таким образом скорректированное отображение пространственной деформации может быть достигнуто с помощью приведенного выше полиномиального преобразования [1].

(11) и (12) представляют собой бинарное квадратное уравнение с 6 параметрами. Как только шесть пар соответствующих точек будут взяты из искаженного изображения и исправленное изображения, оценку параметра (a_i, b_i) можно будет получить путем решения

уравнений. В теории, чем больше пар соответствующих точек взято, тем более точную оценку параметра $(a_i,\ b_i)$можно получить.

Параметр m является соответствующей точкой, поэтому он может быть выражен вектором как:

$$U^i=[u_1,u_2,...,u_m]\quad V^i=[v_1,v_2,...,v_m] \tag{13}$$

$$A=\begin{bmatrix} 1 & x_1 & y_1 & x_1^{\,2} & x_1y_1 & y_1^{\,2} \\ 1 & x_2 & y_2 & x_2^{\,2} & x_2y_2 & y_2^{\,2} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & x_m & y_m & x_2^{\,2} & x_my_m & y_m^{\,2} \end{bmatrix} \tag{14}$$

Коэффициенты матрицы соответственно $a'=[a_0,a_1,...,a_m],b'=[b_0,b_1,...,b_m]$.

Для одного изображения f с координатами пикселя изображения (x,y), пусть после геометрического искажения будет сгенерированно изображение g с координатами пикселя изображения (x',y'). Такое преобразование может быть выражено как:

$$\begin{aligned} x'&=r(x,y)\\ y'&=s(x,y) \end{aligned} \tag{15}$$

Здесь $r(x,y)$ и $s(x,y)$ — пространственные преобразования, которые генерируют изображение с геометрическим искажением $s(x',y')$. Например, если $r(x,y)=x/2$, $s(x,y)=y/2$, искаженное изображение — это просто уменьшение размера $f(x,y)$ наполовину на два пространственных направления.

Если $r(x,y)$ и $s(x,y)$ известны аналитически (численно, то в теории обратное преобразование можно использовать для восстановления $f(x,y)$ по искаженному изображению $g(x',y')$. Однако на практике невозможно сформулировать аналитические функции $r(x,y)$ и $s(x,y)$, и эти аналитические функции описывают геометрические искажения во всей плоскости изображения. Наиболее используемым способом преодоления этой трудности является задача выражения пространственного перемещение пикселя с помощью метода контрольной точки. Эти точки являются подмножеством пикселей, и их позиции на входе (искажение) и выходе (восстановление) изображения точно известны. Пока существует способ найти их соответствующие отношения, и реализовать поправку на каждый пиксель, то можно восстановить всё изображение. Четыре вершины являются соответствующими точками соединения. Предположим, что геометрическая деформация площади четырехугольника моделируется билинейным уравнением, а именно:

$$\begin{aligned} r(x,y)&=c_1x+c_2y+c_3xy+c_4\\ s(x,y)&=c_5x+c_6y+c_7xy+c_8 \end{aligned} \tag{16}$$

$$\begin{aligned} x'&=c_1x+c_2y+c_3xy+c_4\\ y'&=c_5x+c_6y+c_7xy+c_8 \end{aligned} \tag{17}$$

Так как всего известно восемь точек, и эти уравнения могут найти восемь коэффициентов $c_i, i = 1,2,...,8$. Эти коэффициенты формируют модель геометрического искажения, которая используется для преобразования всех пикселей в четырёхугольную область, и такой четырехугольник определяется производным коэффициентом [3].

Билинейный метод интерполяции реализует линейную интерполяция в двух направлениях. Следующие производные формула расчета имеет значение интенсивности серого цвета пикселя, которое необходимо вычислить:

Для $(x, y+v)$ получим:

$$f(x, y+v) = [f(x, y+1) - f(x, y)]v + f(x, y) \tag{18}$$

Для $(x+1, y+v)$ получим:

$$f(x+1, y+v) = [f(x+1, y+1) - f(x+1, y)]v + f(x+1, y) \tag{19}$$

Для $(x+u, y+v)$ получим:

$$\begin{aligned} f(x+u, y+v) &= [f(x+1, y+v) - f(x, y+v)]u + f(x, y+v) = (1-u)(1-v) f(i, j) + (1-u)v f(i, j+1) \\ &+ u(1-v) f(i+1, j) + uv f(i+1, j+1) \end{aligned} \tag{20}$$

Метаданные пациентов и другая медицинская информация хранится в DICOM файлах как объекты с присвоенными характеристиками. [1] Процесс получения DICOM файла представлен на рис. 1.

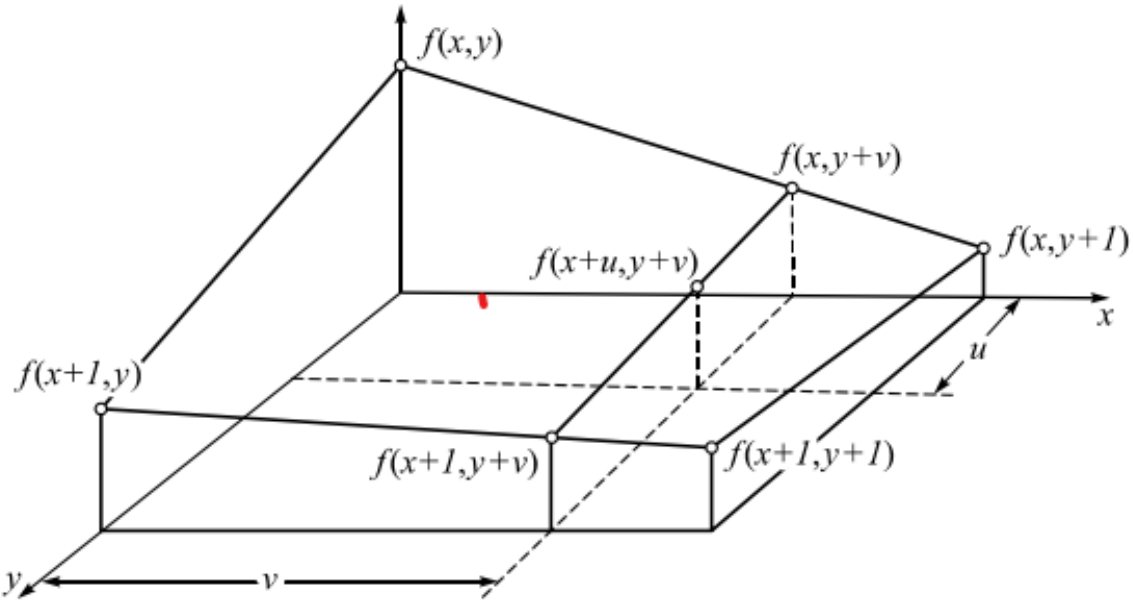


Рис.3. Билинейная интерполяционная схема.

4. Вычислительный эксперимент

Существует множество методов оценки точности восстановления искажённого изображения. В этой статье средняя ошибка и среднее значение корня квадратичной ошибки используется для сравнения эффекта коррекции. Статистические результаты представлены в

табл. 1.

	Направление	Сред. ошибка	Сред.квадратич. ошибка
Метод осн. на гомографии	Горизонтальное	4.26%	126.8225
	Вертикально	3.01%	123.276
Метод с билинейной интерполяцией	Горизонтальное	3.52%	103.472
	Вертикально	3.04%	109.1334

Таблица 1. Сравнение результатов работы методов

Средняя ошибка коэффициента искажения — это скорректированный остаточный коэффициент искажения уже после применения метода для восстановления изображения. Из таблицы видно, что ошибки координат узлов сетки в горизонтальном и вертикальном направлениях улучшены, а алгоритм билинейной интерполяции имеет значительное уменьшение значения ошибки в сравнении с методом основанным на гомографии. Среднеквадратическая ошибка отражает разность значений между вычисляемым положением координаты изображения и фактическим положением координаты изображения. Чем больше разница, тем лучше проходит восстановление.

$$Сред.ошибка = \frac{\sum |f(x_i) - y_i|}{n}, i = 1, 2, 3..., n$$
$$Средн.квадрат.ошибка = \left[\sum_{i=1}^{n} \frac{(f(x_i) - y_i)^2}{n} \right]^{1/2} \quad (21)$$

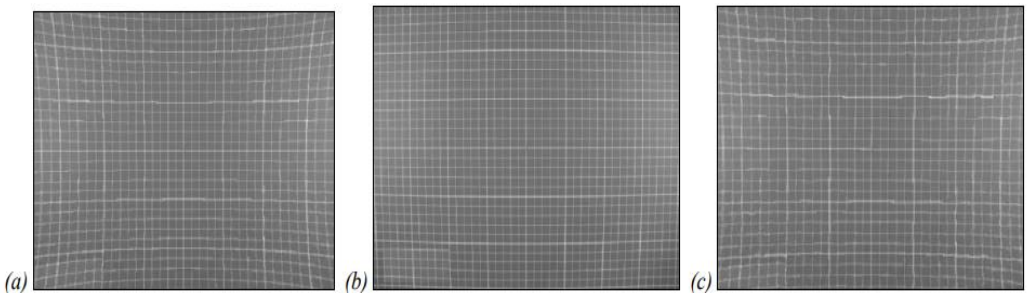


Рис.4. (a) Изображение с искажением (b) Метод осн. на гомографии (c) Метод билин. интерпол.

Заключение

В данной статье были рассмотрены некоторые методы восстановления изображений, искаженных перспективным искажение. Данное исследование помогает получить представление о доступных методах исправление дефектов изображения. Также, в ходе эксперимента был выявлен наиболее эффективный метод восстановления изображения из предложенных.

Работа была выполнена под руководством доцента, канд. физ.-мат. наук, доцента кафедры вычислительной математики и прикладных информационных технологий ВГУ, Медведева Сергея Николаевича и профессора, докт. тех. наук, заведующего кафедрой вычислительной математики и прикладных информационных технологий ВГУ, Леденевой Татьяны Михайловны.

Литература

1. Geoffroy O. Transient climate response in a twolayer energy-balance model. Part I: Analytical solution and parameter calibration using / O. Geoffroy, D. Saint-Martin, DJL. Olivie, A. Voldoire, G. Bellon, S. Tytéca // CMIP5 AOGCM experiments. Journal of Climate – 2013. – 26(6): P. 1841–1857.
2. Geetha Kiran A. Automatic rectification of perspective distortion from a single image using plane homography / A. Kiran Geetha, S. Murali. // International Journal on Computational Sciences & Applications. – 2013. – Vol.3. – No.5. – P. 47– 58.
3. Tan T.L. Contrast enhancement of computed tomography images by adaptive histogram equalization-application for improved ischemic stroke detection / T.L. Tan, K.S. Sim, C.P. Tso, A. K. Chong // International Journal of Imaging Systems and Technology. – 2000. – P. 395–406.
4. Dammann P. Evaluation of hardware-related geometrical distortion in structural MRI at 7 Tesla for image-guided applications in neurosurgery /P. Dammann, O. Kraff, K.H. Wrede, N. Özkan, S. Orzada, O.M. Mueller, I.E. Sandalcioglu, U. Sure, E.R. Gizewski, M.E. Ladd, T. Gasser // Academic Radiology. – 2011. – 18(7). – P. 910–916.
5. Robert A. Volume Rendering / A. Robert, Drebin, L. Carpenter, P. Hanrahan // SIGGRAPH '88. Atlanta. – 1988.
6. Melo R. A new solution for camera calibration and real-time image distortion correction in medical endoscopy-initial technical evaluation / R. Melo, J.P. Barreto, G. Falcao // IEEE Transactions on Biomedical Engineering. – 2012. – 59(3). – P. 634- 644.
7. Xiang W. Geometric and Photometric Correction of Projected Rectangular Pictures / W. Xiang, K. Reinhard, R. Bodo // Image and Vision Computing. – 2006.
8. Yu G. An algorithm for estimation and correction of anisotropic magnification distortion of cryoem images without need of pre-calibration / G. Yu, K. Li, Y. Liu, Z. Chen, Z. Wang, R. Yan, T. Klose, L. Tang, W. Jiang // J Struc Biol. – 2016. – 195(2). – P. 207-215.

УДК 004.42

ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ ДЛЯ АНАЛИЗА ПЛАНОВ ВЫПОЛНЕНИЯ SQL-ЗАПРОСОВ

Е. Р. Ершова

Воронежский государственный университет

Введение

В современном мире объём данных, который необходимо обрабатывать, растёт с каждым днём. В связи с этим всё более важным становится эффективное выполнение SQL-запросов к системе управления базами данных (СУБД).

Цель данной работы – спроектировать приложение, которое будет позволять провести сравнительный анализ планов выполнения запросов различных (СУБД) и определить наиболее эффективный из них. Результаты исследования в дальнейшем будут использованы для оптимизации процесса выполнения запросов и повышения эффективности работы с базами данных.

1. Постановка задачи

Необходимо спроектировать приложение, которое позволяет:

- загружать файл с моделью данных и заполнением;
- создавать базу данных в СУБД;
- выполнять запрос и просмотреть план его выполнения;
- сравнивать планы выполнения запросов.

В качестве систем управления базами данных выбраны PostgreSQL [1], MySQL [2], MS SQL [3], Oracle [4].

2. Схема работы приложения

2.1. Задание настроек

Схема работы приложения представлена на рис. 1.

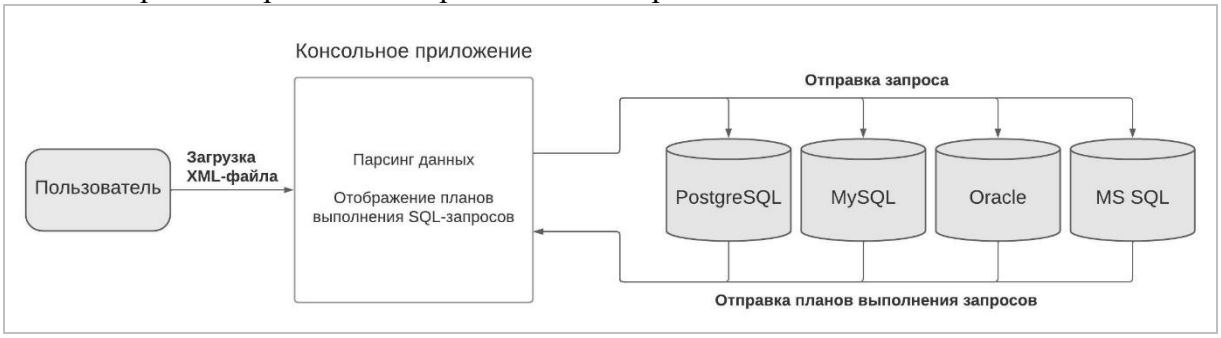


Рис. 1. Схема работы приложения

Пользователь загружает в программу XML-файл, в котором расположены модель

данных и схема генерации данных для заполнения. Структура загружаемого файла отображена на рис. 2.

```
<tables>
    <table tablename >
        <columns>
            <column name, type >
        <columns>
        <primaryKey name, columns >
        <foreignKeys>
            <foreignKey name, baseColumnNames, referencedTableName, referencedColumnNames >
        <foreignKeys>
        <indices>
            <index>
        <indices>
    </table>
</tables>
```

Рис. 2. Структура загружаемого XML-файла

В приложении выполняется парсинг данных. Затем происходит подключение к серверам четырёх СУБД: PostgreSQL, MySQL, MS SQL, SQLite.

2.2. Получение результатов для будущего анализа

К базам данных выполняется запрос, план выполнения которого формируется каждым оптимизатором и в виде XML-документа отправляется в консольное приложение пользователю. С помощью парсинга информация отображается в удобочитаемом виде. Элементы структуры полученных данных выглядят следующим образом:

- Node-Type – название плана сканирования.
- Parallel-Aware – является ли параллельным это процесс.
- Relation-Name – название отношения.
- Alias – название таблицы.
- Startup-Cost – стоимость запуска, то есть время перед началом этапа вывода данных.
- Total-Cost – итоговая стоимость запроса.
- Plan-Rows – ожидаемое число строк, которое должен вывести этот узел плана.
- Plan-Width – ожидаемый средний размер строк, выводимых этим узлом плана (в байтах).
- Actual-Time – фактическое время (в миллисекундах).
- Actual-Rows – число строк, которое вывел этот узел плана.
- Actual-Loops – показывает, сколько всего раз выполнялся этот узел.
- Planning-Time – время, затраченное на построение плана запроса из разобранного запроса и его оптимизацию.
- Execution-Time – время выполнения, которое включает продолжительность запуска и остановки исполнителя запроса, а также время выполнения всех сработавших триггеров, но не включает время разбора, перезаписи и планирования запроса.

Затем необходимо выяснить, какой из оптимизаторов выполнил запрос лучше. Анализ плана выполнения SQL-запроса позволяет найти проблемные участки, которые занимают

большое количество ресурсов и замедляют выполнение запроса.

Эффективность выполнения решено оценивать по следующим критериям [6]:

- время выполнения запроса;
- использование индексов;
- ресурсы, потребляемые запросом (объём оперативной памяти и дискового пространства);
- количество операций с данными;
- степень параллелизма выполнения запроса;
- статистика использования таблиц.

Результаты исследования могут быть использованы в дальнейшем для оптимизации процесса выполнения запросов и повышения эффективности работы с базами данных.

Заключение

В результате работы спроектировано приложение, которое позволяет:

- загружать файл с моделью данных и заполнением;
- создавать базу данных в СУБД;
- выполнять запрос и просмотреть план его выполнения;
- сравнивать планы выполнения запросов.

Научный руководитель: доцент, канд. физ.-мат. наук, доцент кафедры программного обеспечения и администрирования информационных систем ВГУ, Селезнёв Константин Егорович.

Литература

1. PostgreSQL. Официальная документация / The PostgreSQL Global Development Group. – URL: https://www.postgresql.org/docs/ (дата обращения: 02.12.2022).
2. MySQL: Documentation. – URL: https://dev.mysql.com/doc/ (дата обращения: 02.12.2022).
3. Microsoft SQL. Documentation. – URL: https://learn.microsoft.com/en-us/sql/?view=sql-server-ver16/ (дата обращения: 04.12.2022).
4. Oracle Database Documentation. – URL: https://docs.oracle.com/en/database/oracle/oracle-database/ (дата обращения: 05.12.2022).
5. Explain command in PostgreSQL. План выполнения оператора. – URL: https://www.postgresql.org/docs/current/sql-explain.html (дата обращения: 27.01.2023).
6. Winand, M. SQL Performance Explained / M. Winand. – Wien, 2014. – 207 с.

УДК 004.925

ТЕХНОЛОГИИ ПОСТРОЕНИЯ ДОПОЛНЕННОЙ РЕАЛЬНОСТИ

С. А. Есина

Воронежский государственный университет

Введение

Несмотря на то, что это достаточно молодое направление, дополненная реальность (augmented reality, AR) уже используется в самых разных областях. Современные технологии позволяют распространять AR среди широкого круга пользователей.

Принцип дополненной реальности заключается в совмещении виртуальных и настоящих объектов в режиме реального времени. Таким образом, окружающий мир может преобразиться через дополнение виртуальными объектами. Эксперты считают, что в ближайшее время технологии построения дополненной реальности будут развиваться достаточно стремительно.

В основе построения дополненной реальности лежат алгоритмы компьютерного зрения. Вместе со стремительным развитием смартфоном и их вычислительных мощностей становится все более возможным привносить приложения дополненной реальности в различные области человеческой деятельности.

1. Термин «дополненная реальность»

Исследователь Рональд Азума предложил определять дополненную реальность как систему, обладающую следующими свойствами:

- совмещение виртуального и реального миров;
- взаимодействие объектов в режиме реального времени;
- расположение в трехмерном пространстве.

В отличие от виртуальной реальности, где предусмотрено погружение в полностью виртуальную среду, дополненная реальность предполагает именно совмещение окружающей действительности с виртуальными объектами, их интеграцию в жизнь.

2. Построение дополненной реальности

Так как дополненная реальность предполагает внедрение в окружающий мир виртуальных объектов, необходимо однозначно определить место, куда конкретно будет расположен объект[1].

В связи с этим выделяются два принципа определения места для объекта:

- на основании координат местоположения пользователя;
- на основе маркера.

Для определения местоположения пользователя используются специальные датчики, например, акселерометр, гироскоп, магнетометр, GPS-приемник. Местоположение виртуального объекта определяется широтой и долготой. Данный способ построения

дополненной реальности зачастую используется в приложениях, которые применяются на улице или на больших пространствах, потому что точность определения координат с помощью GPS все еще не близка к идеалу.

В случае использования маркеров становится необходимым говорить и об алгоритмах компьютерного зрения. Маркером называется объект, расположенный в окружающем пространстве, который анализируется специальным программным обеспечением для последующей отрисовки 3D-моделей на месте данного маркера.

Далее маркеры распознаются с помощью алгоритмов компьютерного зрения. После того, как маркер найден в видеопотоке и вычислено его местоположение, нужно построить матрицы проекции и позиционирования виртуальных моделей.

3. Анализ алгоритмов компьютерного зрения

За последнее время была создана большая теоретическая база в сфере обработки изображений и поиска на нем различных объектов.

3.1. Метод контурного анализа

Данный метод хорошо подходит для распознавания текста, так как ориентирован на распознавание предметов по их контурам[2].

Контур целиком определяет форму изображения и содержит всю необходимую информацию для распознавания изображений по их форме. Контуром считается кривая, описывающая границу объекта на изображении. Она представляет собой пространственнопротяженный разрыв, перепад или скачкообразное изменение значений яркости[3]. В контурном анализе контур кодируется последовательностью, состоящей из чисел. На контуре фиксируется точка, которая называется начальной. Затем контур обходится (например, по направлению движения часовой стрелки), и каждый вектор смещения записывается, например, комплексным числом $a+ib$, где a — смещение точки по оси x; b — то же по оси y. Смещение берется относительно предыдущей точки.

В библиотеке OpenCV для поиска контуров используется функция cvFindContours(). В ней используется известный способ кодирования контура – цепной код Фримана. Чтобы обнаружить контуры, нужно сначала привести картинку к оттенкам серого, бинаризовать полученное изображение, и после этого воспользоваться функцией cvFindContours().

3.2. Метод сопоставления шаблонов

Этот метод цифровой обработки изображений разработан для поиска небольших частей изображения, соответствующих образу шаблона. Для того чтобы найти на картинке какой-либо объект, необходимо иметь шаблон с изображением этого объекта для поиска на картинке лучшего совпадения.

Выбирается маска, представляющая собой матрицу с координатами каждого пикселя в шаблоне, а затем перемещается по картинке попиксельно[4]. На каждом этапе вычисляется сумма произведений маски и изображения, также представленного в пикселях. Выход перекрестной корреляции будет самым высоким в местах, где структура изображения

соответствует структуре маски, где большие значения изображения умножаются на большие значения маски[5].

В библиотеке OpenCV для реализации этого метода есть функция cvMatchTemplate(). Одним из ее аргументов является способ вычисления корреляции между шаблоном и изображением. В зависимости от выбора значения этого аргумента местоположение заданного шаблона на целевом изображении можно обнаружить по минимуму или максимуму значения на результирующей картинке.

3.3. Метод сравнения ключевых особенностей

Известен также как «feature detection», заключается в вычислении некоторой абстракции изображения с дальнейшим выделением его ключевых особенностей[6]. В дальнейшем выделенные ключевые особенности у двух изображений сравниваются между собой. Однако нет однозначного определения, что такое ключевая особенность, так как в качестве нее могут выступать грани или вершины объектов, кривые или целые связанные области.

В библиотеке OpenCV для реализации этого метода используется абстрактный класс FeatureDetector, в который входят функции detect() и create().

В процессе сравнения ключевых точек можно выделить следующие этапы:

1. Feature detector: поиск ключевых точек на изображении;
2. Descriptor extractor: составление описания найденных ключевых точек с оценкой их позиции через описание окружающих областей;
3. Matcher: построение соответствия между двумя полученными наборами точек изображений;
4. Анализ полученных результатов для определения нахождения требуемого маркера на изображении.

3.4. Генетические алгоритмы

Механизмы генетических алгоритмов поиска напоминают биологическую эволюцию. Реализуются путем случайного подбора, комбинирования и вариации искомых параметров. Генетические алгоритмы компьютерного зрения помогают найти заданных объект на изображении или в видеопотоке. Для обучения алгоритма проводится обучение на «хороших» (содержащих искомый объект) и «плохих» (не содержащих искомый объект) изображениях[7].

При этом для обучения используется большое число изображений, и чем их больше — тем лучше будет работать сам алгоритм. Для каждой картинки производится выделение различных ключевых особенностей: границы, линии, центральные элементы. По ним производится построение статистической модели, которая затем и используется для поиска объекта на изображении.

Примером использования данного подхода может служить алгоритм распознания лиц и глаз на видеопотоке. Постепенно обучая алгоритм, можно добиться высоких результатов нахождения заданного класса объектов. Однако необходимость обучения как раз и делает использование генетических алгоритмов достаточно проблематичным. Для их хорошей работы требуется существенное число различных изображений (как «хороших», так и «плохих»),

и время построения классификатора для каждого объекта может занимать продолжительное время.

Заключение

Простота и быстродействие контурного анализа позволяют вполне успешно применять данный подход при условии наличия четко выраженного объекта на контрастном фоне и отсутствии помех. Наличие одинаковой яркости с фоном и шумы приводят к невозможности выделения контура, а, следовательно, и невозможности распознавания объекта. Также перекрытие объектов или их группировка приводит к тому, что контур выделяется неправильно и не соответствует границе объекта. Исходя из данных замечаний, можно сделать вывод, что такой метод отлично подходит для поиска объекта некоторой заданной формы или для распознавания текста.

Метод сопоставления шаблонов подходит для случаев, когда нужно найти что-то определенное на изображении, которое заранее известно. Он не позволяет с уверенностью сказать, найден ли исходный объект, поскольку это вероятностная характеристика, зависящая от масштаба, углов обзора, поворотов картинки и наличия физических помех. Из-за этого возможны ложноположительные результаты, когда у шаблона и области на тестируемом изображении есть какие-либо схожие фрагменты.

Метод сравнения ключевых особенностей более устойчив к трансформациям и позволяет находить объекты даже при наличии физических помех. Высокая скорость работы некоторых методов feature detection позволяет применять их для поиска изображений в режиме реального времени даже на мобильных устройствах, что способствует использованию дополненной реальности в смартфонах и планшетах.

Генетические алгоритмы представляют собой очень мощный инструмент. Чтобы пользоваться данными алгоритмами, их необходимо долго обучать и загружать в них большое количество «плохих» и «хороших» изображений. При обучении алгоритма можно предусмотреть разные углы поворота объекта, различный масштаб, различные наклоны и т.п. Если их качественно обучить, то данные алгоритмы можно применять для различных задач, например, распознавание лиц, распознавание текста и т.д. Главными недостатками этих алгоритмов являются сложность их реализации и время обучения, а построение классификатора для каждого объекта может занимать достаточно продолжительное время.

Для распознавания маркеров дополненной реальности необходимо, чтобы алгоритм мог распознавать изображения от самых простых до самых сложных. Нужно, чтобы алгоритм считывал маркер даже при поворотах и наклонах. Не все смартфоны обладают хорошей камерой, поэтому необходимо, чтобы алгоритм распознавал объекты с шумами и был быстрым, чтобы можно было считывать маркеры на видеопотоке в режиме реального времени. Алгоритм должен распознавать изображения вне зависимости от их масштаба и быть несложно реализуем.

Таким образом, в качестве продолжения исследования планируется провести практическое тестирование каждого метода на определенных примерах и составить сравнительную характеристику, сравнив ее результаты с теорией, изложенной в данной статье.

Литература

1. Благовещенский И. А. Технологии и алгоритмы для создания дополненной реальности. / И. А. Благовещенский, Н. А. Демьянков // Моделирование и анализ информационных систем – Москва, 2013. – Т. 20, №. 2. – С. 129-138.
2. Bay H. SURF: Speeded up robust features / H. Bay, T. Tuytelaars, L. Van Gool. – Springer Berlin Heidelberg, 2006. – 118 с.
3. Burns J. B. Extracting straight lines. Pattern Analysis and Machine Intelligence / J. Burns, A. R. Hanson, E. M. Riseman. – IEEE, 1986. – 446 с.
4. Mitchell M. An introduction to genetic algorithms. / M. Mitchell. – MIT press, 1998. – 158 с.
5. Canny J. A computational approach to edge detection. / J. Canny. – IEEE, 1986. – 162 с.
6. Башков Е. А. Реалистичная визуализация трехмерных объектов и сцен с использованием технологий объемного отображения / Е. А. Башков, С. А. Зори. – Известия ЮФУ. Технические науки, 2012. – 137 с.
7. Genetic Algorithms Library. - Режим доступа: http://www.aforgenet.com/framework/features/genetic_algorithms.html.– (Дата обращения 20.11.2022).

УДК 519.612

ИССЛЕДОВАНИЕ СХОДИМОСТИ МЕТОДА СОПРЯЖЕННЫХ ГРАДИЕНТОВ ПРИ ПРИМЕНЕНИИ ПРЕДОБУСЛАВЛИВАТЕЛЯ К РАЗРЕЖЕННОЙ СИСТЕМЕ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ

М. В. Жбанкова

Воронежский государственный университет

Введение

На практике часто приходится сталкиваться с задачей решения систем линейных алгебраических уравнений (далее – СЛАУ). Решение разреженных систем больших размеров является важной прикладной задачей во многих промышленных отраслях.

Методы решения СЛАУ можно разделить на прямые и итерационные. Итерационные методы, как правило, применяются при решении задач большой размерности, когда использование прямых методов невозможно из-за ограниченности оперативной памяти и долгого времени счета. Суть итерационного метода состоит в построении последовательности приближений к точному решению СЛАУ. Сходимостью метода называют сходимость данной последовательности к точному решению системы из любого начального приближения.

Однако для того, чтобы применять метод на практике, недостаточно одной его сходимости. В процессе вычислений неизбежно появление вычислительной погрешности. На эффективность решения значительно влияет обусловленность матрицы системы уравнений. Большинство методов быстро сходятся, если матрица хорошо обусловлена или имеет небольшое число собственных значений. В противном случае из-за накопления вычислительной погрешности метод, который сходится в теории, на практике может разойтись. Для борьбы с этой особенностью используется предобуславливание системы – переход к СЛАУ с тем же решением и матрицей, обладающей лучшими качествами, с помощью умножения системы на матрицу специального вида [2–3, 5].

В настоящей работе будет проведен вычислительный эксперимент, демонстрирующий погрешность решения разреженной СЛАУ с помощью одного из популярных итерационных методов решения СЛАУ – метода сопряженных градиентов [2–4]. Данный метод предназначен для решения систем общего вида с невырожденной симметричной положительно определенной матрицей линейных уравнений. На его примере будет показано влияние предобуславливателя на сходимость системы. В качестве предобуславливателя будет рассмотрен предобуславливатель Якоби, который представляет собой диагональную матрицу, элементы которой обратны элементам главной диагонали матрицы исходной системы [1].

1. Постановка задачи и общее описание эксперимента

Задача заключается в проведении вычислительного эксперимента для демонстрации возникающей при использовании метода сопряженных градиентов решения СЛАУ погрешности. Необходимо программно реализовать функции заполнения матрицы системы, матрицы предобуславливателя, функции метода решения с применением предобуславливателя и без него. Эксперимент будет проводиться для систем различной размерности, его суть состоит в следующем:

1. Генерируется разреженная случайная матрица СЛАУ, вектор решений; вектор правой части является результатом умножения матрицы системы на вектор решений.
2. Полученная система решается с помощью метода сопряженных градиентов. Погрешность

вычисляется как разница между сгенерированным вектором решений и полученным в результате работы метода.
3. Дополнительно генерируется матрица обуславливателя. Система решается с помощью предобусловленного метода сопряженных градиентов [1, 2]. Вычисляется погрешность.
4. Проводится сравнение полученных значений погрешности.
По результатам эксперимента можно будет сделать вывод о влиянии предобуславливания на точность решения. Для реализации был использован язык программирования C++.

2. Описание применяемых методов

2.1. Метод сопряженных градиентов

Метод сопряженных градиентов основан на том факте, что решение системы линейных уравнений $\mathbf{Ax}=\mathbf{b}$ с симметричной положительно определенной матрицей $\mathbf{A}$ эквивалентно решению задачи минимизации функции $F(\mathbf{x})=\frac{1}{2}(\mathbf{Ax},\mathbf{x})-(\mathbf{b},\mathbf{x})$. Функция $F(\mathbf{x})$ достигает своего минимального значения тогда и только тогда, когда ее градиент обращается в ноль $\nabla F(\mathbf{x})=\mathbf{Ax}-\mathbf{b}$. Таким образом, решение СЛАУ можно искать как решение задачи безусловной минимизации $F(\mathbf{x})$.

Перед началом итерационного процесса необходимо выбрать начальное приближение $\mathbf{x}_0$, а также $\mathbf{r}_0=\mathbf{b}-\mathbf{Ax}_0$, $\mathbf{z}_0=\mathbf{r}_0$. На рис. 1 приведена реализация основного итерационного процесса на языке C++.

```
int Iteration = 0;
float nev = 0;
do {
    Iteration++;
    if (Iteration != 1) nev = Spr1 / mf;
    Spz = 0;
    Spr = 0;
    for (int i = 0; i < N; i++)
    {
        Sz[i] = 0;
        for (int j = 0; j < N; j++)
        {
            Sz[i] += A[i][j] * Zk[j];
        }
        Spz += Sz[i] * Zk[i];
        Spr += Rk[i] * Rk[i];
    }
    alpha = Spr / Spz;

    Spr1 = 0;
    for (int i = 0; i < N; i++)
    {
        Xk[i] += alpha * Zk[i];
        Rk[i] -= alpha * Sz[i];
        Spr1 += Rk[i] * Rk[i];
    }

    beta = Spr1 / Spr;

    for (int i = 0; i < N; i++)
        Zk[i] = Rk[i] + beta * Zk[i];
} while (Spr1 / mf != nev && Iteration < N);
```

Рис. 1. Функция, реализующая метод сопряженных градиентов

2.2. Предобусловленный метод сопряженных градиентов

Подход предобуславливания заключается в умножении исходной матрицы СЛАУ на матрицу предобуславливателя. Таким образом, вместо решения исходной системы линейных алгебраических уравнений $\mathbf{Ax}=\mathbf{b}$ можно решать предобусловленную слева систему $\mathbf{P}^{-1}\mathbf{Ax}=\mathbf{P}^{-1}\mathbf{b}$, где $\mathbf{P}^{-1}$ – матрица предобуславливателя. При этом произведение $\mathbf{P}^{-1}\mathbf{A}$ не вычисляется явно, так как полученная матрица, скорее всего, будет плотной. Вместо этого в итерационный метод включаются корректирующие шаги, учитывающие предобуславливание. Так как оператор $\mathbf{P}^{-1}$ применяется на каждом шаге итерационного линейного решателя, операция $\mathbf{P}^{-1}$ должна быть легко вычисляемой. Простейший предобуславливатель Якоби (обозначим его $\mathbf{J}$) удовлетворяет этому условию, так как его нахождение является задачей с линейной сложностью: его диагональные элементы $j_{ii}=\frac{1}{a_{ii}}, i=\overline{1,n}$, а все остальные – нули.

Перед началом итерационного процесса необходимо выбрать начальное приближение $\mathbf{x}_0$, а также $\mathbf{r}_0=\mathbf{b}-\mathbf{Ax}_0$, $\mathbf{z}_0=\mathbf{P}^{-1}\mathbf{r}_0$, $\mathbf{p}_0=\mathbf{z}_0$.

На рис. 2 приведена реализация предобусловленного метода сопряженных градиентов.

```
int Iteration = 0;
float nev = 0;
do {
    Iteration++;
    if (Iteration != 1) nev = Spr1 / Nf;
    Spz = 0;
    Spr = 0;

    for (int i = 0; i < N; i++)
    {
        Sz[i] = 0;
        for (int j = 0; j < N; j++)
        {
            Sz[i] += A[i][j] * Pk[j];
        }
        Spr += Rk[i] * Zk[i];
        Spz += Pk[i] * Sz[i];
    }
    alpha = Spr / Spz;

    Spr1 = 0;
    for (int i = 0; i < N; i++)
    {
        Xk[i] += alpha * Pk[i];
        Rk[i] -= alpha * Sz[i];
        Zk[i] = 0;
        for (int j = 0; j < N; j++)
            Zk[i] += M[i][j] * Rk[j];
        Spr1 += Rk[i] * Zk[i];
    }

    beta = Spr1 / Spr;

    for (int i = 0; i < N; i++)
        Pk[i] = Zk[i] + beta * Pk[i];
} while (Spr1 / Nf != nev && Iteration < N);
```

Рис. 2. Функция, реализующая предобусловленный метод сопряженных градиентов

3. Вычислительный эксперимент

Сперва для разреженных систем размерностью от 10 до 500 со случайными элементами найдем решения с помощью метода сопряженных градиентов, далее попробуем применить к этому методу предобуславливатель Якоби. Полученные результаты вычисления погрешности приведены в табл. 1.

Таблица 1

Вычислительные погрешности для случайных разреженных систем различной размерности

Размерность матрицы	Погрешности для метода сопряженных градиентов без применения предобуславливателя	Погрешности для предобусловленного метода сопряженных градиентов
10	1.67017	$6.86646 \cdot 10^{-5}$
20	78.3219	0.000686646
30	93.4854	0.00227356
40	33.2602	0.000152588
50	72.7608	0.000728607
60	29.5536	0.0016861
70	36.2436	0.0205231
80	36.9221	0.0149384
90	30.6835	0.012413
100	60.7518	7.05867
200	19.4863	0.212513
300	26.718	0.397083
400	24.6564	0.202263
500	14.2797	0.0382156

Как можно увидеть, значения погрешностей при решении исходных систем довольно большие, а с применением предобуславливателя решения СЛАУ стали намного точнее, при этом погрешности заметно растут вместе с ростом размерности системы.

Теоретически это объясняется тем, что предобусловленная система более устойчива и обладает меньшим числом обусловленности, чем исходная. Докажем это программно, вычислив число обусловленности для каждой начальной матрицы $\mathbf{A}$ и для произведениия $\mathbf{P}^{-1}\mathbf{A}$. Найденные значения приведены в табл. 2. Числа обусловленности для исходных матриц очень велики, предобуславливание уменьшает эти значения на порядки.

Таблица 2

Число обусловленности матрицы

Размерность матрицы	Случайная разреженная матрица	Произведение предобуславливателя Якоби и случайной разреженной матрицы
10	4097.16	268.744
20	228.807	22.0181
30	1379.09	424.045
40	2243.08	33.9296
50	109143	32.8155
60	624468	205.407
70	15412.4	12166.2
80	5294.14	1753.31

90	5886.47	114.285
100	1223.89	786.823
200	$1.28068 \cdot 10^6$	568221
300	$2.41054 \cdot 10^6$	6414.26
400	$2.78392 \cdot 10^6$	4973.38
500	$2.75118 \cdot 10^6$	39701.6

Заключение

В ходе вычислительного эксперимента было показано влияние предобуславливания на решение случайных разреженных систем линейных алгебраических уравнений различной размерности. Исходные системы имеют очень плохую обусловленность и большие погрешности решения, которые могут быть сравнимы с самими значениями вектора решений. Применение предобуславливателя Якоби уменьшает число обусловленности на порядки и позволяет получить намного более точное решение. Естественно, этого недостаточно для применения на практике там, где требуются высокоточные вычисления, так как хорошо обусловленные системы для обеспечения наибольшей точности должны иметь число обусловленности, близкое к единице, и погрешности решения, близкие к нулю. Предобуславливатель Якоби – один из наиболее простых и очевидных, но при этом помогает заметно улучшить результаты вычислений. Возможно, повышения точности можно добиться, применяя другие, более сложные, виды предобуславливателей и соответствующие иные предобусловленные методы решения СЛАУ.

Литература

1. Hegland, M. Block Jacobi Preconditioning of the Conjugate Gradient Method on a Vector Processor / Markus Hegland, Paul E. Saylor // International Journal of Computer Mathematics. – 1992. – С. 138–157.
2. Saad, Y. Iterative Methods for Sparse Linear Systems / Yousef Saad – Minneapolis: Society for Industrial and Applied Mathematics, 2003. – 446 с.
3. Амосов, А. А. Вычислительные методы для инженеров: учебное пособие / А. А. Амосов, Ю. А. Дубинский, Н. В. Копченова – М.: Высшая школа, 1994. – 544 с.
4. Самарский, А. А. Численные методы / А. А. Самарский, А. В. Гулин. – М.: Наука, 1989. – 430 с.
5. Предобуславливание // Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/Предобуславливание. – (Дата обращения: 24.04.2023).

УДК 004

РАЗРАБОТКА IOS ПРИЛОЖЕНИЯ “FRIENDS” С АРХИТЕКТУРНЫМ ШАБЛОНОМ MVVM

Е. С. Забаштина, С. Ю. Болотова

Воронежский государственный университет

Введение

Целью научной работы является разработка приложения «Friends» под управлением операционной системы iOS. Требовалось разработать масштабируемое приложение с поддержкой всех современных устройств на базе iOS версии 15.0 и выше. Основной характеристикой разрабатываемого приложения являлось использование сервисов и процессов, не зависящих от основного приложения и выполняющих обработку данных в фоновом режиме. Приложение содержит локализации для английского и русского языков, а также поддержку темного режима. Для реализации поставленной задачи был использован открытый мультипарадигменный компилируемый язык программирования общего назначения Swift. Для разработки проекта было решено использовать архитектуру MVVM.

Практическая значимость: изучить и улучшить знания в разработке приложений для мобильных устройств, ознакомиться с многопоточными приложениями и особенностями платформы.

1. Проектирование мобильного приложения с использованием архитектурного паттерна MVVM

Перед проектированием мобильного приложения были разработаны функциональные требования, нашедшие отражение в диаграмме вариантов использования, построенной при разработке концептуальной модели приложения (рис. 1).

Рис. 1. Диаграмма вариантов использования

После описания приложения на концептуальном уровне была разработана его логическая модель, включающая описание основных классов системы и используемых в них полей и методов, а также диаграмму наиболее важных модулей приложения, показанную на рис. 2

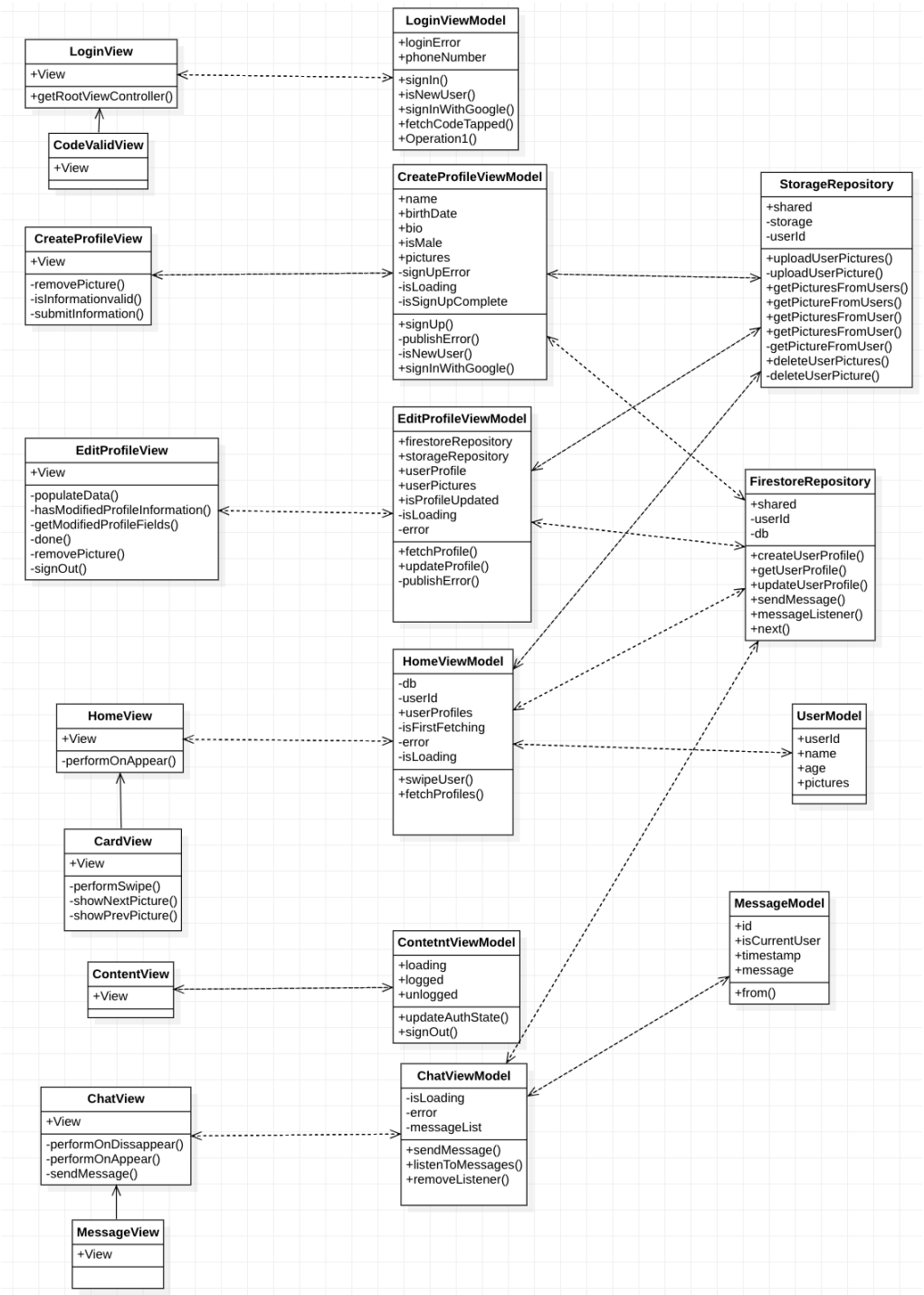


Рис. 2. Диаграмма классов

Мобильное приложение спроектировано на основе шаблона MVVM, который облегчает процесс разработки сложных приложений, отделяет логические части проекта на разные объекты, бизнес-логику от пользовательского интерфейса. Компоненты внутри каждого блока устанавливают свои связи через механизм связывания (binding), который позволяет этим двум сущностям взаимодействовать совершенно прозрачным образом. Связывание является мостом между View и ViewModel.

Приложение разработано с помощью фреймворка SwiftUI, который имеет декларативный стиль написания кода. «Friends» соответствует принципам дизайна Apple, а именно обеспечивает согласованность дизайна пользовательского интерфейса.

Приложение состоит из следующих основных экранов:

- LoginView.

Разрешен вход через Google или номер телефона. Если у пользователя есть учетная запись, он будет перенаправлен на домашнюю страницу, в противном случае появится экран «Создать профиль»

- CreateProfileView.

Пользователю потребуется выполнить следующие действия:

- Добавить как минимум две фотографии профиля, которые можно получить через галерею мобильного телефона или камеру устройства. Соответственно запрашиваются необходимые разрешения.
- Указать имя пользователя.
- Указать дату рождения для расчета возраста.
- Указать пол (для упрощения получения профиля доступны только два варианта).
- Указать биографию. Данное поле не является обязательным и может содержать до 500 символов. Оставшееся количество символов отображается по мере ввода пользователем.
- Указать один вариант категории времяпрепровождения.

После того, как информация будет заполнена и пользователь нажмет кнопку «Зарегистрироваться», если пользователь не существовал ранее и создание учетной записи было успешным, пользователь будет перенаправлен на домашнюю страницу, в противном случае появится диалоговое окно ошибки.

- HomeView.

Здесь пользователь может просматривать категории встреч. При переходе на экран выбранной категории ему предлагаются профили, которые выбрали данный формат встречи. Если пользователю нравится пользователь, то он может смахнуть вправо, тогда он будет перенаправлен на ChatView, в котором можно отправить сообщение. Чат активен 48 часов, за это время пользователи должны договориться о встрече или диалог будет удален. Если пользователю не понравился другой пользователь, то он может его смахнуть влево, и он больше не будет показан этому пользователю.

Из HomeView пользователь может получить доступ к экрану редактирования профиля и экрану сообщений.

- EditProfileView.

На этом экране пользователь может изменять те же поля, что и на экране «создать профиль», за исключением имени и даты рождения. Их конструкция практически идентична.

- MessageView.

Здесь пользователь может увидеть свои диалоги и получить доступ к соответствующему экрану чата, чтобы отправлять другим пользователям сообщения.

- ChatView.

Здесь пользователь может отправлять сообщения, и они будут обновляться в режиме реального времени.

Заключение

В результате исследования был описан проект разработки нативного мобильного приложения «Friends». Практическим результатом исследования является рабочее мобильное приложение, протестированное на мобильных устройствах под управлением операционной системы iOS.

Литература

1. Official Site Swift. [Electronic resource] — URL: https://docs.swift.org/swift-book/
2. Усов В. А. «Swift. Основы разработки приложений под iOS, iPadOS и macOS». 6-е изд. дополненное и переработанное, 2021. — 544 с.
3. Смит Н. «SwiftUI Essentials - iOS 14 Edition. Learn to Develop iOS Apps using SwiftUI, Swift 5 and Xcode 12», 2020. — 492 с.
4. Чейрд В. «Swift подробно». 2020. — 412 с.
5. Нахавандипур В. «iOS. Приемы программирования». 2014 — 832 с.
6. Мартин Р. «Чистый код. Создание, анализ и рефакторинг». 2019 — 464 с.
7. Конспект лекций Стэнфордского университета «Разработка iOS приложений» 2021. [Electronic resource] — URL: https://bestkora.com/IosDeveloper/ios-14-swiftui-2/

УДК 004.942

ИССЛЕДОВАНИЕ И РЕАЛИЗАЦИЯ ЗАДАЧИ НЕЛИНЕЙНОЙ ОПТИМИЗАЦИИ С ИСПОЛЬЗОВАНИЕМ МЕТОДА КОНЕЧНЫХ ЭЛЕМЕНТОВ ДЛЯ СИМУЛЯЦИИ ПОВЕДЕНИЯ КОСТЕЙ И ПЛОТИ ПРИ АКТИВАЦИИ МЫШЦ

А. С. Забияко, Е. В. Трофименко

Воронежский государственный университет

Введение

Метод конечных элементов [1] на сегодняшний день является наиболее популярным способом решения инженерных задач механик твёрдого тела, замкнутых газов и жидкостей. В данной работе мы рассмотрим реализацию задачи нелинейной оптимизации [2] с использованием метода конечных элементов для решения задачи симуляции поведения плоти и смещения костей при активации мышц.

Реализация подобного метода симуляции даёт возможность решить сразу 3 класса востребованных во многих сферах задач:

– задачу по нахождению деформации плоти и смещения костей при активации мышц;

– задачу по нахождению активации мышц, зная деформацию плоти и смещение костей;

– задачу по нахождению деформаций плоти и смещения костей путём переноса мышечных активаций из другой модели.

В данной работе мы поэтапно рассмотрим все необходимые детали, которые необходимо учесть при реализации такого метода симуляции.

1. Построение задачи

1.1. Симуляция плоти

Для того чтобы симулировать поведения плоти мы будем использовать метод конечных элементов для изотропного твёрдого тела. В первую очередь, для работы с методом конечных элементов, необходимо разбить деформируемый объект на конечное количество элементов. Так как наша симуляция будет происходить в 3-ех измерениях, то мы будем разбивать модель плоти на тетраэдры.

Теперь необходимо выбрать энергию деформации, которая будет аппроксимировать поведение реальной плоти. Считается что человеческая плоть при небольших деформациях практически несжимаема, сохраняет свой объем и стремится к изначальной форме. Для аппроксимации такого поведения будем использовать стабильную неогуковскую энергию на основе деформационных градиентов [3]. Запишем её формулу в терминах инвариантов:

$$\Psi^i_{\text{SNH}} = \frac{\mu}{2}(I_2^i - 3) - \mu(I_3^i - 1) + \frac{\lambda}{2}(I_3^i - 1)^2, \quad (1)$$

где i *— индекс тетраэдра плоти,*

μ *— первый параметр Ламе,*

λ *— второй параметр Ламе,*

I_2^i *— инвариант, показывающий изменение суммы квадратов длин рёбер тетраэдра,*

I_3^i *— инвариант, показывающий изменение объёма тетраэдра.*

Для удобства эти параметры можно выразить через модуль Юнга E (физическая величина, характеризующая способность материала сопротивляться растяжению, сжатию при упругой деформации) и коэффициент Пуассона v (величина отношения относительного поперечного сжатия к относительному продольному растяжению):

$$\mu = \frac{E}{2(1+\nu)}, \tag{2}$$

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}. \tag{3}$$

Для аппроксимации поведения человеческой плоти будем использовать коэффициент Пуассона $\nu = 0,49$, что аппроксимирует сохранение объёма при деформациях и модуль Юнга $E = 0{,}05$, что аппроксимирует упругость плоти.

1.2. Симуляция мышц

Для симуляции воздействия мышц на плоть существует множество различных способов, начиная от введения внешних сил, воздействующих на плоть и заканчивая симуляцией отдельных групп мышечных волокон. В данной работе мы будем реализовывать один из простейших и вычислительно эффективных методов, на основе деформаций тетраэдров плоти в соответствии с активацией мышц [4].

Суть метода заключается в том, что сначала мы находим долю объёма, который занимает мышца в каждом из тетраэдров плоти. Затем каждый тетраэдр плоти деформируем с коэффициентом полученной доли в соответствии с активацией мышцы.

Для того чтобы получить формулу энергий активаций мышц нам необходимо знать деформационные градиенты и повороты тетраэдров плоти в которые попала мышца:

$$E^{i,j}_{active} = V^{i,j}_{active}(\mu \left\| \mathbf{F}^i - \mathbf{R}^i\mathbf{S}^j \right\|^2_F + \lambda(\det(\mathbf{F}^i) - \det(\mathbf{S}^j))^2), \tag{4}$$

где j *— индекс мышцы,*

$V^{i,j}_{active}$ *— объём* j *мышцы в* i *тетраэдре,*

$\mathbf{F}^i$ *— деформационный градиент тетраэдра,*

$\mathbf{R}^i$ *— матрица поворота тетраэдра,*

$\mathbf{S}^j$ *— матрица активации мышцы.*

Учитывая объём мышцы выпишем новую формулу для энергии плоти:

$$E^{i,j}_{passive} = (V^i - V^{i,j}_{active})\Psi^i_{\text{SNH}}, \tag{5}$$

где V^i *— объём тетраэдра.*

1.3. Симуляция костей

Так как кости являются твёрдыми недеформируемыми объектами в рамках небольших деформаций, то в данной работе мы ограничимся нахождением смещения и поворота костей при активации мышц и прикреплением костей к плоти.

Так для того, чтобы получить смещение и поворот для кости, достаточно ввести в оптимизацию соответствующие свободные переменные и добавить к ним ограничивающую функцию в оптимизацию:

$$E^k_{transform} = w^k_r \left\| \bar{\mathbf{r}}^k \right\|^2_F + w^k_t \left\| \bar{\mathbf{t}}^k \right\|^2_F, \tag{6}$$

где k *— индекс кости,*

w_r^k *— вес степени свободы вращения кости,*

w_t^k *— вес степени свободы смещения кости,*

$\bar{\mathbf{r}}^k$ *— вектор поворотов кости,*

$\bar{\mathbf{t}}^k$ *— вектор смещения кости.*

Для прикрепления плоти к кости достаточно выделить точки на плоти, которые мы хотим прикрепить и зафиксировать их относительно полигонов кости, например, с помощью барицентрических координат. Чтобы это сделать введём ещё одну ограничивающую функцию, которая с учётом поворотов и смещения кости будет сохранять положения прикреплённой плоти:

$$E_{fit}^p = w_{fit}^p \left\| (\bar{\mathbf{v}}_b^p)^T - \mathbf{M}^k (\bar{\mathbf{v}}_t^p)^T \right\|_F^2, \tag{7}$$

где p *— индекс крепления,*

w_{fit}^p *— вес степени свободы крепления,*

$\bar{\mathbf{v}}_b^p$ *— координаты вершины плоти, которую мы прикрепляем,*

$\mathbf{M}^k$ *— матрица трансформации кости, полученная из свободных переменных* $\bar{\mathbf{r}}^k$ и $\bar{\mathbf{t}}^k$,

$\bar{\mathbf{v}}_t^p$ *— координаты точки на кости, к которой мы прикрепляем плоть.*

1.4. Финальная функция ошибки

Общая функция ошибки для всех элементов по которой будет происходить оптимизация будет иметь следующий вид:

$$E = \sum_{i,j} E_{passive}^{i,j} + \sum_{i,j} E_{active}^{i,j} + \sum_k E_{transform}^k + \sum_p E_{fit}^p . \tag{8}$$

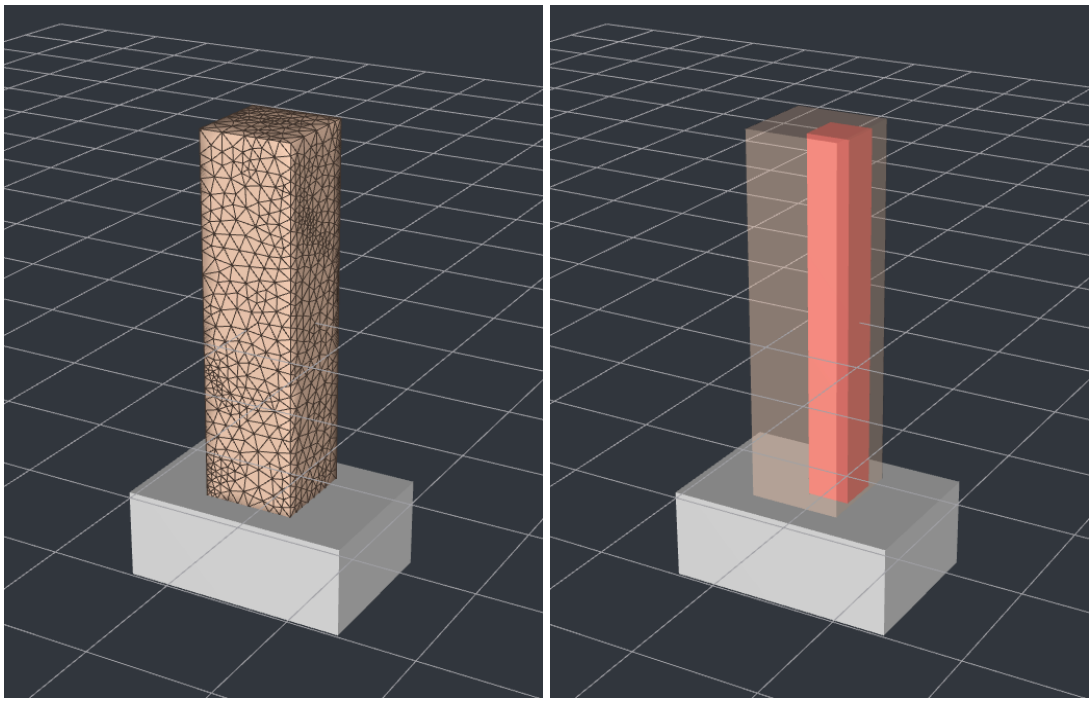
Свободными переменными оптимизации у нас будут вершины тетраэдров плоти, повороты тетраэдров в которые попали мышцы, а также смещения и повороты костей.

Стоит заметить, что полученная формула не учитывает соединения между костями, а также возможные самопересечения плоти и пересечения плоти с костями, что является довольно большой темой само по себе.

2. Реализация

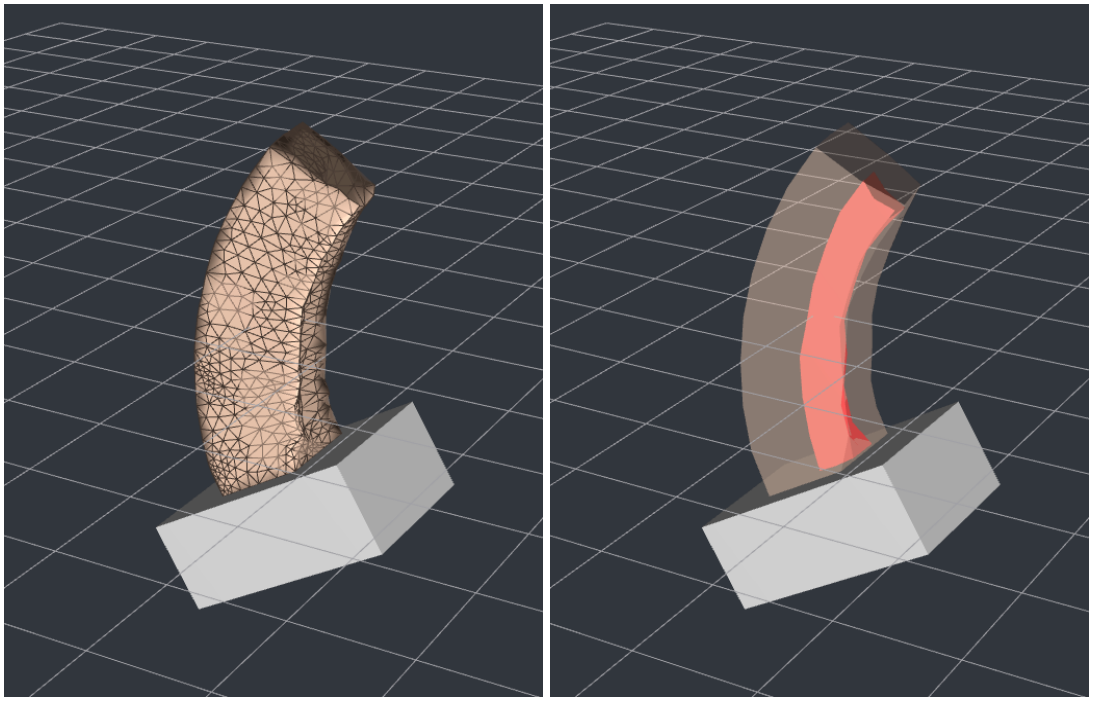
Для реализации полученной ранее задачи нелинейной оптимизации был использован язык программирования C++, так как он позволяет добиться высокой скорости вычислений и использует минимальное потребление оперативной памяти. При разбиении объектов на тетраэдры была использована библиотека TetGen [5]. Для быстрого вычисления матричных операций была использована библиотека с открытым исходным кодом Eigen [6]. Фреймворк с открытым исходным кодом CasADi [7] использовался для автоматического дифференцирования выражений и создания задачи нелинейно оптимизации. Для решения такой задачи была использована библиотека IPOPT [8], реализующая итеративный алгоритм метода внутренней точки. Интерфейс приложения был разработан с использованием фреймворка Qt [9]. Для визуализации объектов симуляции с помощью графического процессора был использован OpenGL.

Для тестирования написанного приложения проведём симуляцию деформации участка плоти с мышцей, прикреплённого одной стороной к кости в форме параллелепипеда. Результаты работы симуляции представлены на рисунках 1, 2 и 3. На рисунках визуально видно, что симуляция деформации плоти и смещения кости под воздействием активации мышцы работает корректно.



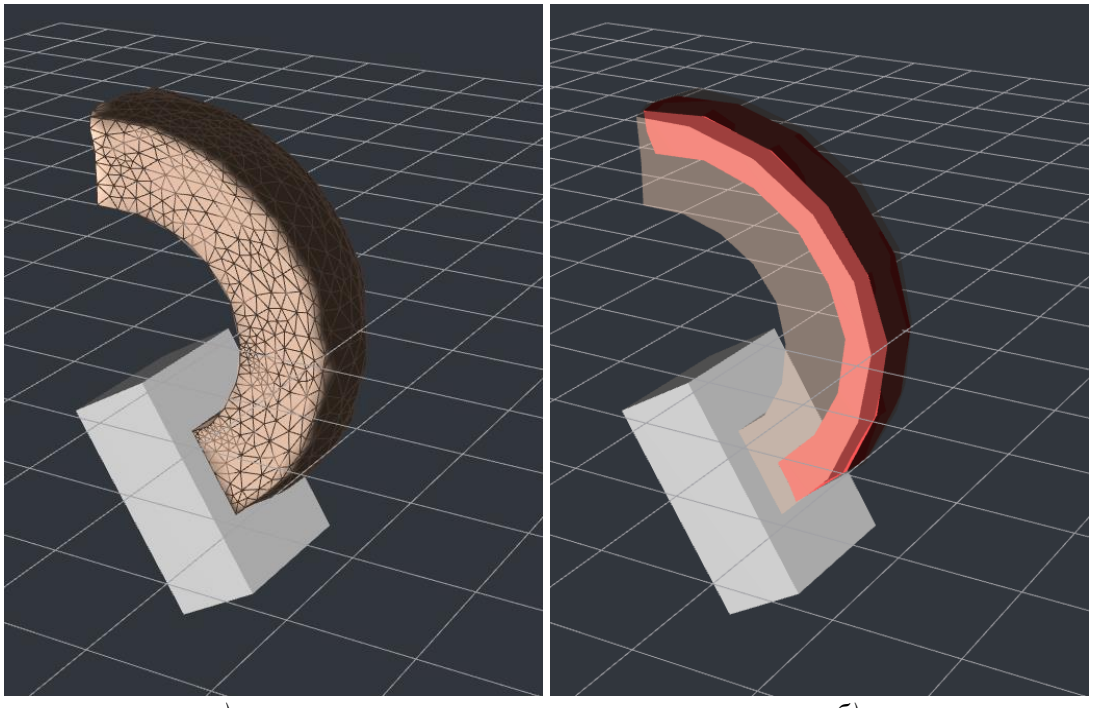
а) *б)*

Рис. 1. Модель плоти, кости и мышцы до деформации: а) полигональная сетка; б) прозрачное отображение



а) *б)*

Рис. 2. Модель плоти, кости и мышцы при сжатии мышцы в 2 раза: а) полигональная сетка; б) прозрачное отображение



а) *б)*

Рис. 3. Модель плоти, кости и мышцы при растяжении мышцы в 2 раза: а) полигональная сетка; б) прозрачное отображение

Заключение

В ходе данной работы были изучены и рассмотрены все основные этапы, необходимые для того, чтобы реализовать симуляцию поведения плоти, костей и мышц, используя метод конечных элементов. Было написано приложение на языке программирования C++ с использованием дополнительных фреймворков, позволяющее вычислять и визуализировать результат работы симуляции. Были продемонстрированы результаты работы симуляции на примере куска плоти, мышцы и кости при сжатии и растяжении мышцы.

Также используя метод построения нелинейной оптимизации для симуляции поведения костей и мышц, описанный в данной работе можно реализовать обратную задачу оптимизации. В таком варианте имея два набора моделей, состоящих из одинакового набора плоти, костей и мышц, но имеющих разную форму деформации плоти, можно найти такой набор активаций мышц, который бы приводил форму плоти одной модели к форме плоти другой.

Литература

1. Eftychios D. Sifakis. FEM Simulation of 3D Deformable Solids: A practitioner’s guide to theory, discretization and model reduction. / Eftychios D. Sifakis, Jernej Barbič // Course, SIGGRAPH. – 2012. – 50 с.

2. Коннов, И. В. Нелинейная оптимизация и вариационные неравенства : учебник для бакалавров / И. В. Коннов. – Казань : Издательство Казанск.ун-та, 2013. – 508 с.

3. Alexandru - Eugen Ichim. Phace: Physics-based Face Modeling and Animation / Alexandru - Eugen Ichim, Petr Kadleček, Ladislav Kavan, Mark Pauly // ACM Transactions on Graphics – 2017. – 14 с.

4. Theodore Kim. Dynamic Deformables: Implementation and Production Practicalities / Theodore Kim, Yale University, David Eberle // Course, SIGGRAPH – 2020. – 188 с.

5. TetGen : официальный сайт. – Режим доступа: https://wias-berlin.de/software/index.jsp?id=TetGen&lang=1 (дата обращения: 12.04.2023).

6. Eigen : официальный сайт. – Режим доступа: https://eigen.tuxfamily.org/index.php?title=Main_Page (дата обращения: 14.04.2023).

7. CasADi : официальный сайт. – Режим доступа: https://web.casadi.org/ (дата обращения: 14.04.2023).

8. IPOPT : официальный сайт. – Режим доступа: https://coin-or.github.io/Ipopt/ (дата обращения: 16.04.2023).

9. Qt Framework : официальный сайт. – Режим доступа: https://www.qt.io/product/framework (дата обращения: 18.04.2023).

УДК 004.932

УДАЛЕНИЕ ПЕРИОДИЧЕСКОГО ШУМА С ИЗОБРАЖЕНИЙ В ЧАСТОТНОЙ ОБЛАСТИ

И. О. Завьялова

Воронежский государственный университет

Введение

Современный мир трудно представить без цифровых изображений. Камеры смартфонов, видеорегистраторы, камеры наружного наблюдения, видеокамеры в жилых домах и магазинах – всё это окружает нас в повседневной жизни и является источником разнообразной информации, объемы которой постоянно увеличиваются.

Однако вне зависимости от происхождения этой информация и её свойств, задача получения максимально точного сигнала остается неизменной. Но вследствие искажения сигнала различными шумами произвести точную регистрацию данных не представляется возможным. Причиной возникновения данных шумов могут быть как внешние условия, так и внутренние, например, отклонения параметров регистрирующих устройств.

Искажения могут не вносить значимых изменений в изображения. Но в некоторых случаях, если качество изображений особенно важно, даже небольшие шумы способны оказывать негативное воздействие. Именно поэтому задача шумоподавления на сегодняшний день является одной из самых распространенных и актуальных в области цифровой обработки изображений.

Целью статьи является рассмотрение результата применения метода фильтрации в частотной области для подавления периодического шума на изображении.

1. Введение в Фурье-анализ

Частотные методы улучшения изображений основаны на выполнении преобразования Фурье над функцией двух переменных – функцией дискретного изображения [1]. Прямое дискретное Фурье-преобразование функции $f(x,y)$ изображения размером $M \times N$ задается равенством

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-2i\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)},$$

где $u = 0,1,...,M-1$, $v = 0,1,...,N-1$.

Обратное преобразование Фурье определяется выражением

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{2i\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)},$$

где $x = 0,1,...,M-1$, $y = 0,1,...,N-1$.

Переменные u и v называются переменными преобразования или частотными переменными, а переменные x и y называются пространственными переменными или переменными изображения. Как правило, числа M и N являются четными для упрощения

компьютерной реализации, а центр Фурье-образа находится в точке с координатами: $u = \frac{M}{2} + 1$, $v = \frac{N}{2} + 1$ [2]. Значение Фурье-преобразования в точке $(u, v) = (0, 0)$ равно

$$F(0,0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y).$$

Таким образом, если $f(x, y)$ – изображение, то значение Фурье-преобразования в начале координат равно среднему значению яркости на изображении. Поскольку началу координат отвечают нулевые частоты, то величину $F(0,0)$ часто называют постоянной составляющей спектра. В силу того, что функция $f(x, y)$ вещественная, спектр Фурье-преобразования изображения обладает свойством симметрии.

Справедливы следующие соотношения между отсчетами в пространственной и частотной области: $\Delta u = \frac{1}{M \Delta x}, \Delta v = \frac{1}{N \Delta y}$.

Фурье-спектр изображения состоит из пикселей, имеющих большой динамический диапазон яркости. Система воспроизведения изображений, как правило, не способна правильно отобразить столь большой диапазон значений интенсивности, что приводит к тому, что при обычном отображении спектра Фурье теряется значительное число деталей. В этой связи для улучшения зрительного восприятия полутонов изображение спектра подвергается логарифмическому преобразованию. Для того чтобы центрировать спектр, необходимо исходное изображение умножить на $-1^{(x+y)}$ перед вычислением Фурье-преобразования.

2. Основы частотной фильтрации

Частотная область цифрового изображения представляет собой не что иное, как пространство, в котором принимают значения переменные (u, v) Фурье-преобразования. Как известно, частота сигнала прямо связана со скоростью изменения сигнала, поэтому интуитивно понятно, что частоты в Фурье-преобразовании связаны с вариацией яркости на изображении. Наиболее медленно меняющаяся (постоянная) частотная составляющая $(u = 0, v = 0)$ совпадает со средней яркостью изображения. Низкие частоты, соответствующие точкам вблизи начала координат Фурье-преобразования, определяют медленно меняющиеся компоненты изображения. По мере удаления от начала координат более высокие частоты начинают соответствовать все более и более быстрым изменениям яркости, представляющим собой границы объектов и другие детали изображения, характеризующиеся резкими изменениями яркости, например, шум на изображении.

Процедура фильтрации изображения в частотной области состоит из следующих шагов:

1. Исходное изображение умножается на $-1^{(x+y)}$, чтобы его Фурье-преобразование оказалось центрированным.

2. Вычисляется прямое дискретное преобразование Фурье (ДПФ) $F(u, v)$ исходного изображения.

3. Функция $F(u, v)$ умножается на функцию фильтра $H(u, v)$.

4. Вычисляется обратное ДПФ от результата шага 3.

5. Выделяется вещественная часть результата шага 4.

6. Результат шага 5 умножается на $-1^{(x+y)}$.

Функция фильтра $H(u, v)$ или передаточная характеристика фильтра подавляет некоторые частоты преобразования, оставляя другие при этом без изменения.

На стадии предварительной обработки, помимо умножения изображения на $-1^{(x+y)}$, могут применяться операции яркостного масштабирования, нормировка размеров исходного изображения, преобразование формата входных данных в формат с плавающей точкой и ряд других.

3. Основные частотные фильтры и их свойства

Основными видами используемых фильтров являются: низкочастотный фильтр, ослабляющий высокие частоты, одновременно пропуская низкие; высокочастотный фильтр, обладающий противоположными свойствами. Низкие частоты Фурье-преобразования отвечают за возникновение превалирующих значений яркости на гладких участках изображения, в то время как высокие частоты отвечают преимущественно за контуры и шум. После применения низкочастотной фильтрации изображение по сравнению с исходным изображением содержит меньше резких деталей. После применения высокочастотной фильтрации на изображении уменьшаются изменения яркости в пределах больших гладких областей и выделяются переходные зоны быстрого изменения яркости, то есть контуры изображения [3]. Как правило, такое изображение обладает большей резкостью по сравнению с исходным.

Другим видом частотного фильтра является фильтр-пробка, или режекторный фильтр, вырезающий определенное значение яркости из изображения. Как правило, это значение яркости в точке начала координат – среднее значение яркости изображения. Среднее значения яркости изображения не может быть строго равно нулю, поскольку для этого некоторые элементы изображения должны содержать отрицательные значения, а средства отображения графической информации не могут оперировать с отрицательными значениями яркости. Для устранения указанного противоречия наименьшее отрицательное значение приравнивают к нулю (уровень черного), а остальные значения пропорционально увеличивают.

3.1. Сглаживающие частотные фильтры

Как отмечалось ранее, контуры и другие резкие перепады яркости на изображении, включая шумы, вносят значительный вклад в высокочастотные составляющие его Фурье-преобразования. Сглаживание изображения в частотной области достигается путем ослабления высокочастотных компонент определенного диапазона Фурье-образа данного изображения.

Модель фильтрации изображения в частотной области в обобщенном виде может быть описана следующим равенством: $G(u,v) = H(u,v) \cdot F(u,v)$, где $F(u,v)$ – Фурье-образ изображения, которое подлежит операции фильтрации, $H(u,v)$ – передаточная функция фильтра, которая ослабит высокочастотные компоненты $F(u,v)$ и сформирует функцию $G(u,v)$.

4. Подавление периодического шума

Подавление периодического шума выполняется с помощью фильтрации в частотной области. Периодический шум на спектре изображения проявляется в виде импульсно подобных всплесков.

Основной метод борьбы основан на режекторной фильтрации – фильтрации, которая не пропускает сигналы (световые колебания) с частотами некоторого определенного диапазона и пропускает сигналы со всеми другими частотами. Узкополосные режекторные фильтры не пропускают частоты в некоторых окрестностях своих центральных частот.

Рассмотрим два вида фильтров для удаления периодического шума: идеальный фильтр и фильтр Баттерворта.

Передаточная функция узкополосного режекторного фильтра Баттерворта порядка n имеет вид:

$$H(u,v) = \frac{1}{1+\left[\frac{D(u,v)W}{D^2(u,v)-D_0^2}\right]^{2n}},$$

где $D(u,v) = \sqrt{(u-\frac{M}{2})^2+(v-\frac{N}{2})^2}$ – расстояние от точки (u,v) до начала координат $(\frac{M}{2},\frac{N}{2})$, W – ширина кольца, D_0 – радиус окружности, проходящей через его середину.

Передаточная функция идеального кольцевого режекторного фильтра задается выражением:

$$H(u,v) = \begin{cases} 1, D(u,v) < D_0 - \frac{W}{2} \\ 0, D_0 - \frac{W}{2} \le D(u,v) < D_0 + \frac{W}{2} \\ 1, D(u,v) > D_0 + \frac{W}{2} \end{cases},$$

где $D(u,v)$ – расстояние, измеряемое от центра частотного прямоугольника, D_0 – радиус окружности, проходящей через середину кольца, W – ширина кольца.

5. Программная реализация

В качестве средства разработки использовался язык Python 3 с использованием программного обеспечения Anaconda 3. Код, используемый для данной работы, был написан в интерактивной веб-оболочке Jupyter Notebook.

Для решения поставленных задач были реализованы следующие функции.

Для перехода из пространственной области в частотную была написана функция ***shift***, которая принимает на вход матрицу ***pix***, хранящую значения яркости пикселей изображения, и возвращает матрицу такого же размера, начало координат которой для Фурье-преобразования находится в точке с координатами $(u = \frac{height}{2}, v = \frac{width}{2})$, где $height$ – высота входного изображения, а $width$ – его ширина.

Для вычисления двумерного дискретного преобразования Фурье написана функция ***DPF*** (Листинг 1), выполняющая как прямое двумерное дискретное преобразование Фурье (тогда параметр ***forward***, подающийся на вход этой функции, должен быть равен единице), так и обратное (тогда параметр ***forward*** равен нулю). Также на вход этой функции в качестве аргумента подается матрица ***matrix***, для которой и находится двумерное ДПФ.

Листинг 1

```
def DPF(matrix, forward):
    new_matrix = DPF_rows(matrix,forward) # находим ДПФ по строкам
    new_matrix = DPF_rows(new_matrix.transpose(), forward)
    new_matrix = new_matrix.transpose()
    return new_matrix
```

Данная функция обращается к функции ***DPF_rows***, выполняющей одномерные дискретные преобразования Фурье всех строк матрицы ***matrix***. Аналогично предыдущей функции данные преобразования могут быть прямыми (***forward*** равен единице) или же

обратными (***forward*** равен нулю). Кроме того, для реализации двумерного ДПФ использовалась функция ***DPF_vect***, вычисляющая одномерное ДПФ подаваемого на вход в качестве параметра вектора ***vector***. В зависимости от параметра ***forward*** оно может быть прямым (***forward*** равен единице) или обратным (***forward*** равен нулю).

Для подавления периодического шума были реализованы функции для двух фильтров: идеального режекторного фильтра и режекторного фильтра Баттерворта порядка n. Функция ***IdealFilter_matrix*** возвращает матрицу идеального режекторного фильтра с параметрами W – ширина кольцевой полосы вокруг начала координат, в которой подавляются частоты, и D_0 – радиус окружности, проходящей через середину кольцевой полосы.

Применение идеального режекторного фильтра реализовано посредством функции ***Ideal_RegectorFilter*** (Листинг 2).

Листинг 2

```
def Ideal_RegectorFilter(matrix, W, D0):
    width, height = matrix.shape[1], matrix.shape[0]
    H = IdealFilter_matrix(W,D0)
    for u in range(height):
        for v in range(width):
            matrix[u][v] *= H[u][v]
    return matrix
```

Функция ***ButterworthFilter_matrix*** возвращает матрицу режекторного фильтра Баттерворта порядка n с параметрами W – ширина кольцевой полосы вокруг начала координат, в которой подавляются частоты, и D_0 – радиус окружности, проходящей через середину кольцевой полосы. Применение этого фильтра реализовано посредством функции ***Butterworth_RegectorFilter*** (Листинг 3).

Листинг 3

```
def Butterworth_RegectorFilter(matrix, W, D0, n):
    width, height = matrix.shape[1], matrix.shape[0]
    H = ButterworthFilter_matrix(W,D0,n)
    for u in range(height):
        for v in range(width):
            matrix[u][v] *= H[u][v]
    return matrix
```

6. Результаты

Рис. 1. Исходное изображение с периодическим шумом

На рис. 1 представлено изображение, сильно искаженное периодическим шумом, использующееся для тестирования программы.

Выведем Фурье-спектр данного изображения в частотной области. Для этого проделаем первые два шага алгоритма фильтрации изображения в частотной области, а именно, переход из пространственной области в частотную с помощью функции ***shift*** и вычислим прямое дискретное преобразование Фурье посредством функции ***DPF***, а затем выведем изображение полученного результата. На рис. 2, где представлен Фурье-спектр входного изображения, хорошо видны частотные компоненты шума в виде пар симметричных

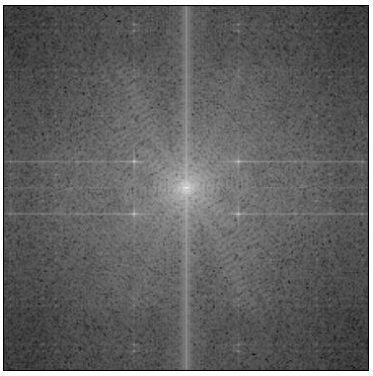


Рис. 2. Фурье-спектр исходного изображения

ярких точек. В данном случае эти компоненты лежат приблизительно на окружности с центом в начале координат частотного пространства, и, таким образом, использование центрально-симметричного режекторного фильтра представляется вполне оправданным.

На рис. 3 представлен идеальный режекторный фильтр с параметрами $W = 11, D_0 = 30$, которые выбраны таким образом, чтобы шумовые импульсы полностью попадали в соответствующую область. Применим к полученному на втором шаге Фурье-спектру данный идеальный режекторный фильтр и продолжим выполнение алгоритма фильтрации изображения в частотной области, вычислив обратное дискретное преобразование Фурье и выполнив переход из области частотных переменных в область пространственных.

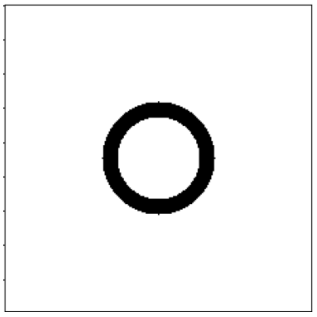


Рис. 3. Идеальный режекторный фильтр $W = 11, D_0 = 30$

Результат фильтрации входного изображения с помощью выбранного фильтра представлен на рис. 4. Улучшение изображения вполне очевидно. Использованный фильтр позволил эффективно восстановить детали изображения, однако посредством данной фильтрации шумы были устранены не полностью, а само изображение стало немного размытым.

Рис. 4. Результат применения идеального режекторного фильтра

Попробуем применить к исходному изображению режекторный фильтр Баттерворта, с параметрами $n = 4, W = 79, D_0 = 34,5$ (рис. 5). Область действия данного фильтра тоже выбрана так, чтобы шумовые импульсы Фурье-преобразования полностью попадали в его область. Результат применения данного фильтра показан на рис. 6. Можно заметить, что данная фильтрация полностью не устранила периодический шум на изображении (он всё ещё наблюдается по краям), однако в целом изображение стало намного чётче и детальнее, стали видны контуры и очертания.

Таким образом, можно сделать вывод, что, применяя режекторные фильтры при подавлении периодического шума на изображении, желательно удалять как можно меньшую часть Фурье-преобразования, тщательно подбирая параметры фильтра таким образом, чтобы область его действия совпадала с шумовыми импульсами, которые и создают периодический шум.

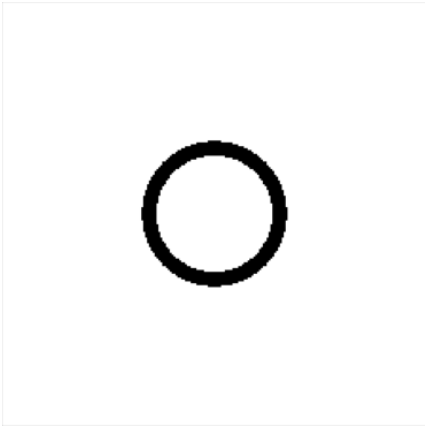


Рис. 5. Режекторный фильтр Баттерворта
$n = 4, W = 79, D_0 = 34,5$

Рис. 6. Результат применения режекторного фильтра Баттерворта

Заключение

В результате проделанной работы были описаны и наглядно продемонстрированы методы подавления периодического шума на изображении в частотной области. Очевидно, что рассмотренные методы показывают достаточно хорошие результаты при правильном подборе параметров.

Литература

1. Гонсалес, Р. С. Цифровая обработка изображений / Р. С. Гонсалес, Р. Е. Вудс. – 3-е изд., исправл. и доп. – Москва : Изд-во Техносфера, 2012 – 1104 с.

2. Загарян, Ю. А. Компьютерная графика в практических приложениях / Ю. А. Загарян, Е. В. Загарян, – Таганрог : Изд-во ТТИ ЮФУ, 2009 – 255 с.

3. Цисарж, В. В. Математические методы компьютерной графики / В. В. Цисарж, Р. И. Марусик. – Киев : Изд-во Факт, 2004. – 464 с.

УДК 519.21; 65.012.323

УПРАВЛЕНИЯ ТРУДОВЫМИ РЕСУРСАМИ ПРОЕКТА В СЛУЧАЕ ЭКСТРЕННОГО ПЕРЕПЛАНИРОВАНИЯ

А. Н. Зобова, Булгакова И.Н.

Воронежский государственный университет

Введение

В современном бизнесе проектное управление является ключевой компетенцией для успешного завершения проектов и достижения бизнес-целей. Однако, при идентификации рисков срыва сроков завершения проекта, возможно экстренное перепланирование, что может привести к увеличению затрат и нарушению бюджета проекта. В такой ситуации оптимизация стоимости проекта на основе человеческого капитала может стать решающим фактором для успешного завершения проекта в ограниченные сроки и с снижением затрат.

В целом, оптимизация стоимости проекта является сложной задачей, требующей комплексного подхода и учета множества факторов. Ее успешное выполнение может значительно повысить эффективность проекта и улучшить его конечный результат.

1. Факторы, приводящие к необходимости пересмотра графика работ

Экстренное перепланирование — это подход к планированию, который широко используется в условиях неопределенности. Он позволяет вносить изменения в график выполнения работ в ответ на внешние и внутренние события.

Причинами экстренного перепланирования проектов могут являться:

1. *Изменение требований клиента*. Изменение требований клиента может привести к изменению объема работ, сроков и бюджета проекта. Это может вызвать необходимость пересмотра плана проекта и его перепланирования, что повлечет за собой дополнительные затраты на бюджет и время проекта.
2. *Непредвиденные обстоятельства*. Непредвиденные обстоятельства, такие как непогода, технические проблемы или человеческие ошибки, могут привести к задержкам в выполнении работ и срыву сроков проекта. Это может вызвать необходимость перепланирования проекта, чтобы справиться с задержками и сократить влияние этих обстоятельств на проект.
3. *Недостаточные ресурсы*. Недостаточное количество ресурсов, таких как деньги, персонал или оборудование, может привести к нехватке ресурсов для выполнения задач проекта в срок. Это может привести к необходимости перепланирования проекта, чтобы управлять ресурсами более эффективно и справиться с задержками.
4. *Недостаточная оценка рисков* Недостаточная оценка рисков может привести к возникновению проблем в проекте, которые могут повлиять на сроки и бюджет проекта. Если эти риски не были учтены на предыдущих этапах проекта, то перепланирование может понадобиться, чтобы управлять этими рисками и снизить их влияние на проект.

Последствия экстренного перепланирования проекта могут быть значительными и включают в себя задержки в сроках, увеличение затрат на проект, снижение качества и ухудшение отношений с клиентами. Если перепланирование не будет проведено эффективно, то это может привести к неудовлетворительному результату проекта.

2. Трудовые ресурсы в проекте

Как упоминалось ранее, трудовые ресурсы одни из основных ресурсов в большинстве проектов.

Человеческий капитал [5] — это термин, который относится к знаниям, навыкам, опыту и таланту людей, которые участвуют в проекте. Он описывает ценность, которую сотрудники могут добавить в проект в результате своих индивидуальных способностей и возможностей для развития. Человеческий капитал является важным фактором в проектном менеджменте, поскольку качество и эффективность проекта зависит от знаний и опыта людей, работающих над ним. Кроме того, управление человеческим капиталом включает в себя развитие и мотивацию сотрудников, чтобы они могли максимально эффективно использовать свои знания и навыки в проекте.

Для формализации определения человеческого капитала будем использовать термин «функциональные возможности» (ФВ), введенный компанией «McKinsey» в статье [6]. Предлагается рассматривать группы специалистов, которые имеют одинаковые функциональные способности для решения бизнес-задач, как ключевые элементы в решении этих задач. Учитывая широкое распространение различных форм временного использования человеческих ресурсов, таких как аутсорсинг и фриланс, концепция ФВ включает не только постоянных сотрудников компании, но также фрилансеров и сотрудников, работающих на условиях аутсорсинга.

Существует несколько способов снижения затрат на проект связанные с человеческим капиталом:

1. *Оптимизация расписания работы*. Распределение рабочих часов сотрудников может быть организовано более эффективно, чтобы уменьшить время, потраченное на проект. Например, можно использовать методологии Agile или Kanban, которые помогают снизить время, потраченное на согласование задач и управление проектом.
2. *Оптимизация рабочих процессов*. Автоматизация повторяющихся задач и оптимизация рабочих процессов может существенно уменьшить количество времени, которое сотрудники тратят на работу. Например, использование софта для автоматического выполнения рутинных задач может ускорить работу и сократить время, затраченное на выполнение проекта.
3. *Обучение и развитие сотрудников*. Предоставление обучения и развития сотрудникам может увеличить их производительность и квалификацию, что в свою очередь может ускорить выполнение проекта и сократить затраты на его реализацию.
4. *Перевод на удаленную работу*. Перевод на удаленную работу может снизить затраты на проект, поскольку позволяет уменьшить расходы на аренду офисных помещений и коммунальные услуги, а также уменьшить транспортные расходы сотрудников.
5. *Использование услуг фрилансеров.* Вместо того чтобы нанимать на постоянную работу сотрудников, можно использовать услуги фрилансеров. Это может быть более дешевым вариантом, особенно если проект требует дополнительных специализированных навыков, которых недостаточно на данный момент у постоянных сотрудников.

3. Стратегия управления трудовыми ресурсами

Рассмотрим стратегию, в соответствии с которой предприятие поддерживает некоторую постоянную $M_ф$ численность ФВ за счёт штатных сотрудников, а сверх этого уровня потребности покрывается путем привлечения внешних ресурсов (фрилансеров и аутстафферов). Преимущество таких внешних ресурсов состоит в возможности мгновенного

привлечения и отказа от продолжения работ без дополнительных затрат, недостаток - в более высокой текущей стоимости ресурса.

Целью реализации такой стратегии является возможность экономии на оперативно неиспользуемых ресурсах при неравномерной потребности в них.

Неравномерность потребности с течением времени будет изменяться в зависимости от независимых одинаково распределенных величин ε_t (потребности в сотрудниках) с функцией распределения вероятности $P(\varepsilon)$.

Эффект от ФВ определяется как:

$$V = (M_{cp}v - c_{\text{соц}}) - M_{\text{ф}}\widehat{c_{\text{шт}}} - c_{out}\sum_{\xi=M_{\text{ф}}}^{\infty}(\varepsilon - M_{\text{ф}})P(\varepsilon), \quad (1)$$

где c_{out} — затраты на временного специалиста в течение определенного интервала времени, $M_{cp} = \sum_{\varepsilon=0}^{\infty}\varepsilon N(\varepsilon)$ — средняя потребность, $c_{\text{соц}}$ — общие затраты на персонал (без учета конкретного сотрудника), включают в себя расходы на создание и поддержание организации, оплату труда и стоимость социальных льгот и услуг, а также другие смежные расходы.

Время работы сотрудника может быть представлено случайной экспоненциально распределенной величиной, т.е. $\pi_i^-(\Lambda,\theta)$ не зависит от стажа (θ), и $\pi_i^-(\Lambda,\theta) = \pi_i^-(\Lambda) == \pi_0^- \approx \frac{1}{10}\cdots\frac{1}{13}$.

Вероятность принятия будущим сотрудником предложения о работе составляет:

$$\pi_i^+(\Lambda) = \pi_0^+ \approx 0{,}95.$$

Параметр Λ отражает оценку важности работы в компании для каждого сотрудника по сравнению с альтернативными вариантами на рынке труда и его целесообразно заменить на $c_{\text{зп}}$. $c_{i.l}^{\text{зп}}(\theta,t)$ — компенсационный пакет и налоги на него.

Зависимость обеих вероятностей от затрат на компенсационный пакет $\pi^-(c_{\text{зп}})$ и $\pi^+(c_{\text{зп}})$ является результатом социально-психологических факторов и не может быть точно оценена, измерена или объяснена объективными законами. Однако, размышления на основе здравого смысла позволяют сформулировать некоторые свойства:

$$\text{при } c \gg c_{\text{зп.0}}; \pi^-(c) \approx 0, \pi^+(c) \approx 1;$$

$$\text{при } c \ll c_{\text{зп.0}}; \pi^-(c) \approx 1, \pi^+(c) \approx 0;$$

$$\pi^-(c_{\text{зп.0}}) = \pi_0^- \text{ и } \pi^+(c_{\text{зп.0}}) = \pi_0^+.$$

Затраты на организационного сотрудника определяются как:

$$\widehat{c_{\text{шт}}} = c_{\text{зп}}\left(1 + \alpha_{\text{оф}}\frac{\pi^-(c_{\text{зп}})}{\pi^+(c_{\text{зп}})} + \alpha_{\text{пр}}\pi^-(c_{\text{зп}})\right).$$

Тогда задача (1) связана с минимизацией затратной части:

$$Z(M_{\text{ф}}) = \alpha_{out}M_{\text{ф}} + \sum_{\varepsilon=M_{\text{ф}}}^{\infty}(\varepsilon - M_{\text{ф}})P(\varepsilon) \to \min; \quad (2)$$

$$\text{где } \alpha_{out} = \widehat{c_{\text{шт}}}/c_{out}.$$

Возможно ее численное решение, так как функция $\sum_{\varepsilon=M_{\text{ф}}}^{\infty}(\varepsilon - M_{\text{ф}})P(\varepsilon)$ монотонно убывает с ростом $M_{\text{ф}}$, и оптимум легко находится простым перебором значений $M_{\text{ф}}$.

Построение выборочной функция распределения вероятностей $P(\xi)$ выполняется на основе исторических данных, необходимых для решения задачи (2).

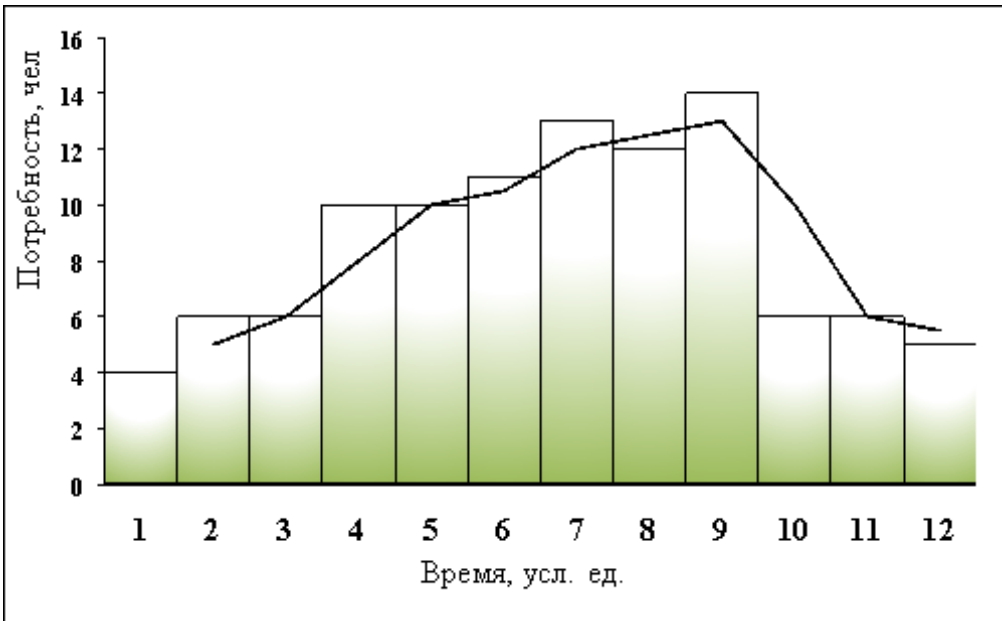


Рис.1. Выборочная потребность в сотрудниках в зависимости от времени

При низкой стоимости внешних специалистов по сравнению со штатными специалистами ($\alpha_{out} \cong 1;\ \widehat{c_{\text{шт}}} = c_{out}$) функция затрат $Z(M_{\text{ф}})$ не имеет минимума: принимает постоянное значение, равное среднему значению потребности.

При высокой стоимости внешних специалистов ($\alpha_{out} = 0, \ll 1;\ c_{out} = 10\widehat{c_{\text{шт}}}$) точка минимума лежит ближе к верхней границе диапазона возможных значений потребности, что объяснимо, и минимум сильно выражен.

При наличии исторических данных и достоверности предположения о сохранении в будущем существующих тенденций применения выборочной функции распределения вероятностей является наиболее адекватным и простым способом её получения оптимальных значений стоимости штатных сотрудников $M_{\text{Ф}}$.

При отсутствии таких данных, например, в условиях создания новых бизнесов и ФВ или при ожидании значительных изменений внешних или внутренних факторов, функция $P(\varepsilon)$ может быть аппроксимирована распространенными распределениями, что позволит получить приблизительные оценки значений $M_{\text{Ф}}$.

Заключение

Управление человеческим капиталом является важной темой, поэтому целесообразно развивать эту область в различных направлениях. Предложенная методика позволяет формировать практические рекомендации по управлению человеческим капиталом через управление оптимальными вариантами затрат на персонал и оценкой ценности ФВ предприятия, которое создает и интенсивно использует знания.

Литература

1. Белов М.В. Модель управления человеческим капиталом / М. В. Белов // Проблемы управления. – 2016. – № 5. – С. 24-34.
2. Боронина Л.Н. Основы управления проектами: учебное пособие / Л. Н. Боронина - Екатеринбург: Изд-во Урал. ун-та, 2015. - 112 с.
3. Добрынин А. И. Человеческий капитал в транзитивной экономике: формирование, оценка, эффективность использования / А. И. Добрынин, С.А. Дятлов, Е. Д. Цыренова. – СПб.: Наука, 1999. – 309 с.
4. Мясоедова Т. Г. Человеческий капитал и конкурентоспособность предприятия / Т. Г. Мясоедова // Менеджмент в России и за рубежом. – 2005. – №3. – С. 29-30.

УДК 004.4

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ТЕСТИРОВАНИЯ ШКОЛЬНИКОВ ПО ОСНОВАМ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Н. А. Зубова, В. П. Железняк

Воронежский государственный университет

Введение

В настоящее время особо остро стоит проблема защиты информации, поэтому необходимо начинать рассказывать об основах информационной безопасности со школьной скамьи.

В настоящее время современная система образования наиболее активно внедряет использование информационных технологий и сети Интернет в образовательном процессе. Быстрее всего развивается переход на электронные системы контроля обучения, например, электронные журналы и дневники, а также активно используются электронные книги и различные онлайн-тестирования.

Актуальность проблемы безопасности человека в информационном обществе определяется, с одной стороны, большим количеством изменений в информационном мире, с другой стороны, необходимостью общества быстро перестраиваться и привыкать к стремительной информатизации. Массовая осведомлённость о потенциальных угрозах и способах защиты информации, начиная со школьной скамьи, поможет построить грамотное информационное общество. Именно поэтому первым шагом на пути к такому обществу стала разработка приложения, которое поможет провести тестирование школьников на предмет основ информационной безопасности.

1. Анализ функциональности приложения

Перед началом разработки был проведён сравнительный анализ существующих решений. В результате анализа были выявлены следующие проблемы: у существующих решений либо отсутствует встроенное тестирование по информационной безопасности либо тест не имеет возможности сохранения результатов и возможности редактирования вопросов.

Требования к необходимой функциональности программы были сформулированы следующим образом: необходимо разработать регистрацию и авторизацию пользователя, для пользователей с учётной записью ученика необходимо разработать выбор темы и теста, а также просмотр результатов, для пользователей с учётной записью учителя необходимо разработать создание и редактирование тестирования и просмотр результатов учеников.

На рис. 1 представлена необходимая функциональность в виде диаграммы вариантов использования.

Так же в ходе анализа была определена и описана модель данных. О пользователе хранится следующая информация: фамилия, имя, отчество, логин и пароль, принадлежность к определённому классу, году обучения. Также хранится информация о темах обучения, лекциях и их содержании, части тестирования с вопросами и ответами, рейтинге пользователя по результатам прохождения тестирования. Связь между классами и темами обучения и тестирования раскрывается дополнительно, с помощью вспомогательной таблицы.

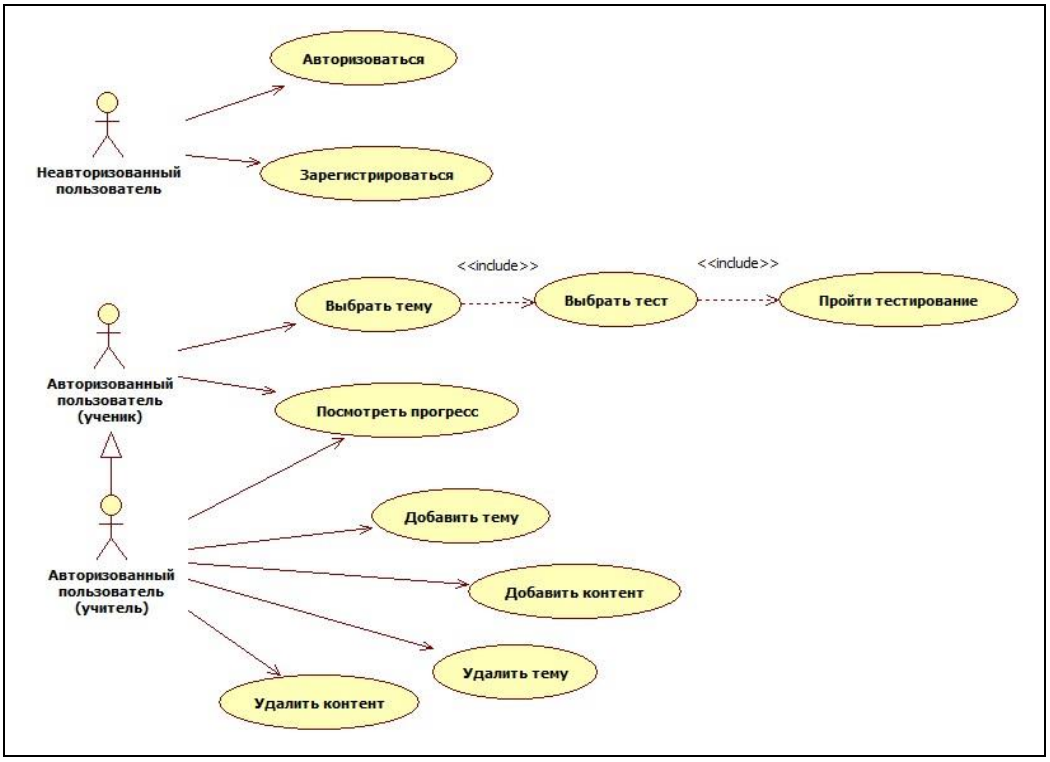


Рис. 1. Диаграмма вариантов использования

2. Средства реализации

При реализации проекта использовались следующие программные средства:

1. Среда разработки Microsoft Visual Studio 2022.
2. Язык разработки C# и платформа для создания интерфейса WPF.
3. Реляционная система управления базами данных Microsoft SQL Server Management Studio 18.
4. Программный инструмент моделирования WhiteStarUML.

3. Описание интерфейса

Приложение реализовано с помощью платформы для создания интерфейса WPF. В виде страничного приложения.

Страница авторизации в приложении представлена на рис. 2.

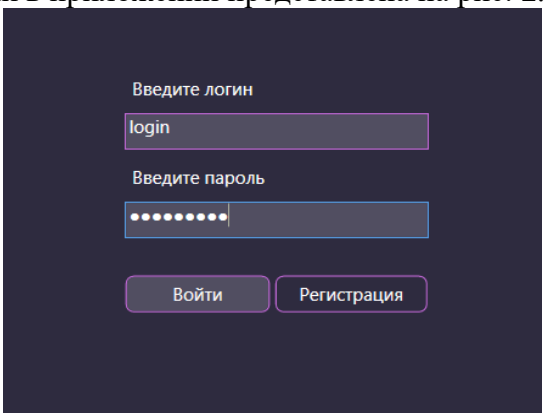


Рис. 2. Окно авторизации

В случае, если пользователь ещё не зарегистрирован в приложении, необходимо перейти к странице регистрации (рис. 3). После авторизации в приложении пользователь выбирает в меню раздел тестирования. После чего выбирает тему и название теста, который необходимо пройти (рис. 4).

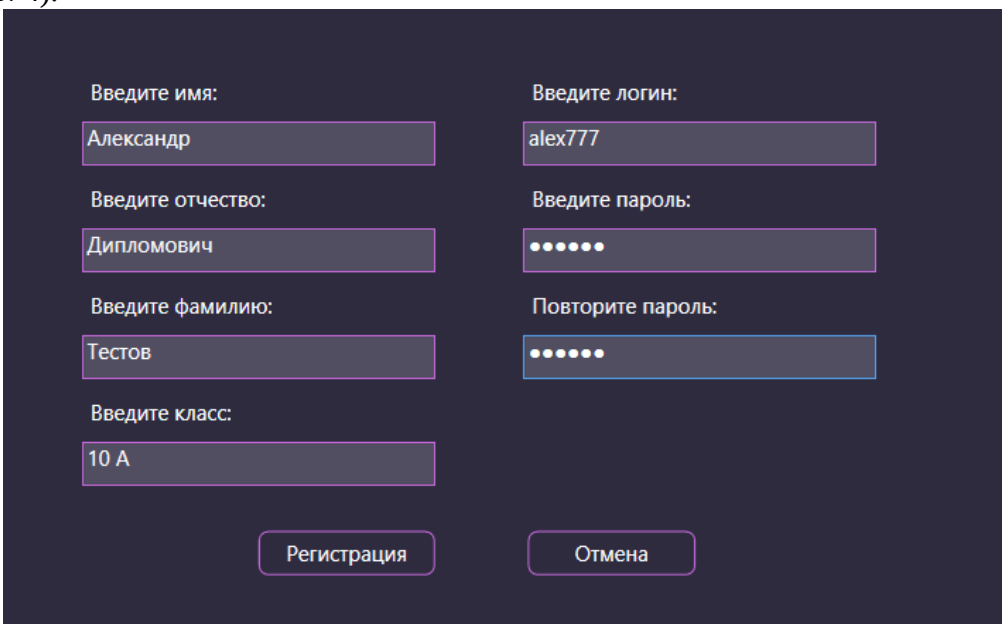


Рис. 3. Регистрация пользователя

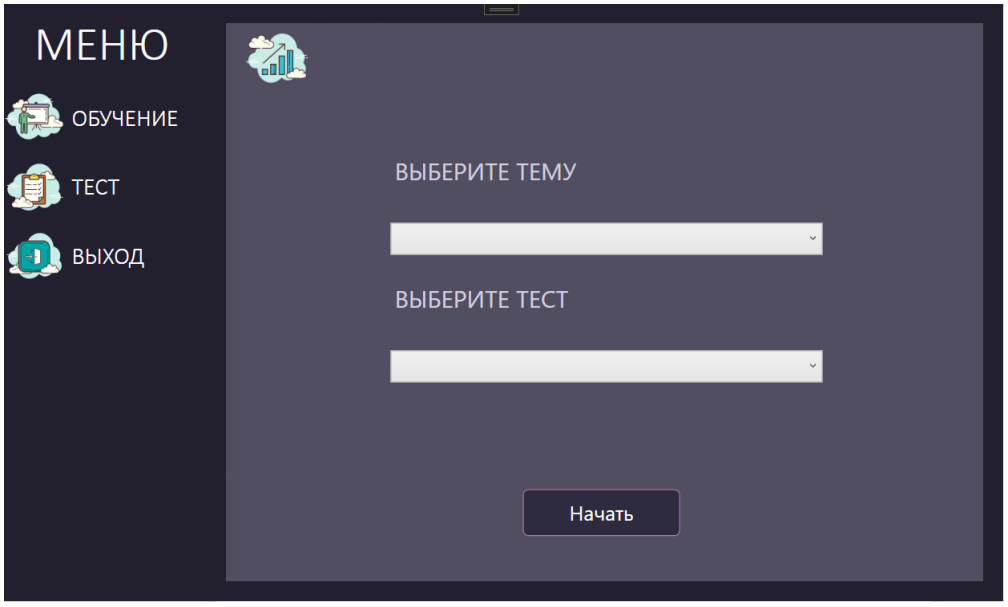


Рис. 4. Выбор тестирования

После выбора темы тестирования пользователю открываются вопросы теста (рис. 5). По окончании прохождения теста ученик получает результаты тестирования (рис. 6).

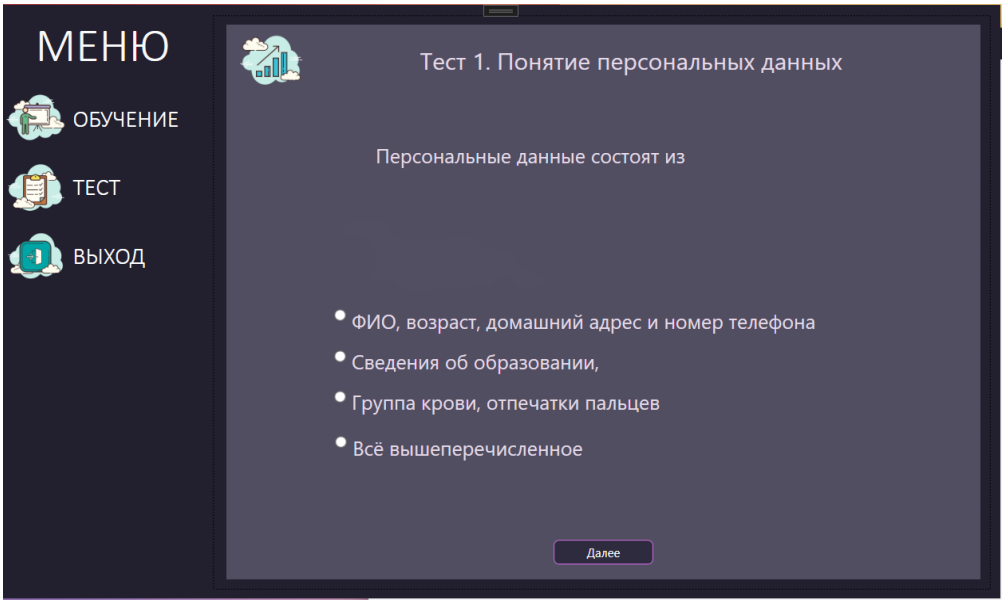


Рис. 5. Окно содержания теста

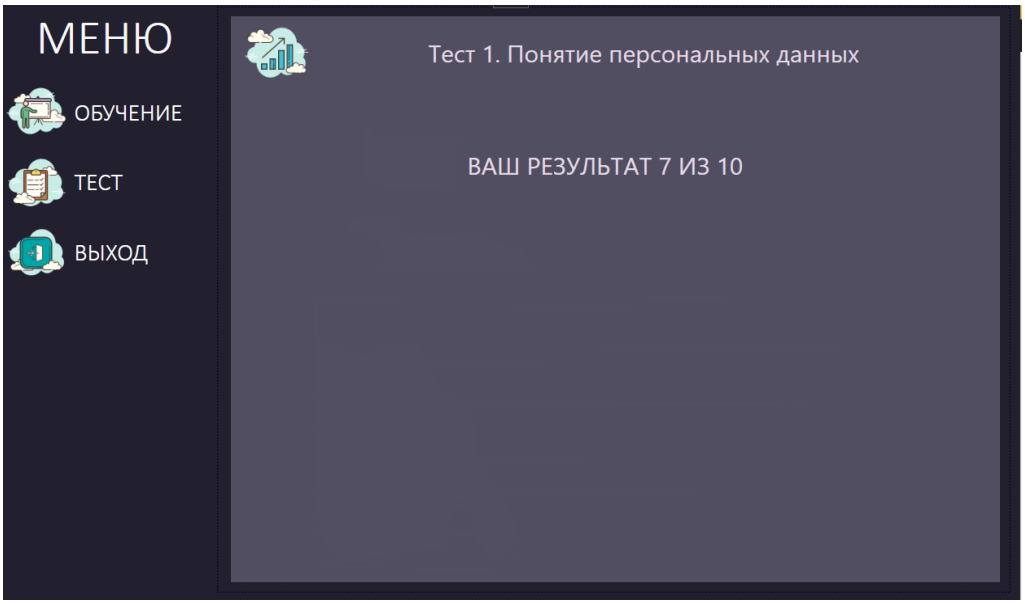


Рис. 6. Результаты тестирования

4. Реализация

В приложении по части тестирования основную функциональность несёт класс CheckTest, который непосредственно реализует проверку ответов и подсчёт правильных ответов. Деятельность класса представлена на диаграмме (рис. 7).

Чтобы обеспечить защиту пароля, в приложении использовано хеширование с помощью встроенного класса System.Security.Cryptography. Вычисляется хеш MD5 для исходных данных ComputeHash, с помощью экземляра класса MD5CryptoServiceProvider. Для авторизации введёный пользователем пароль аналогично хешируется и проводится сравнение двух хеш-значений.

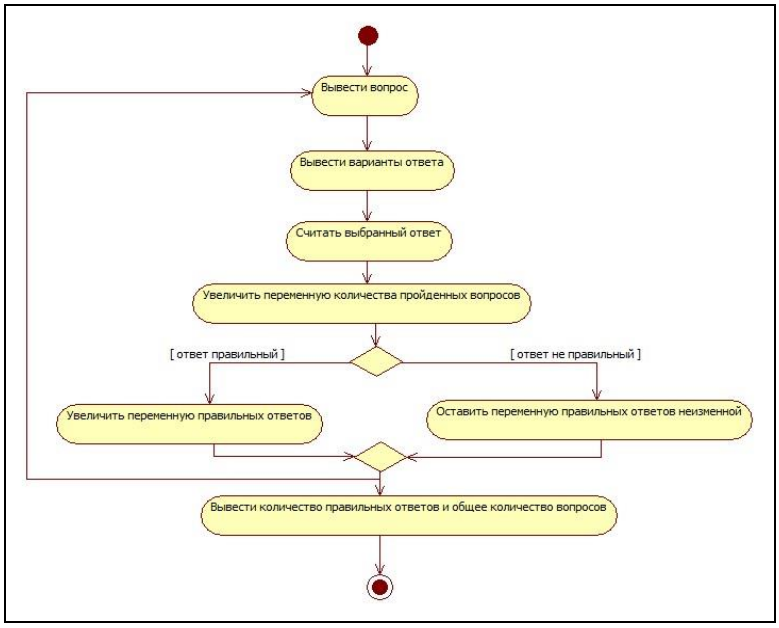


Рис. 7. Диаграмма деятельности

Заключение

Разработанное приложение уже обладает достаточно большой функциональностью относительно аналогичных решений, а являясь абсолютно бесплатным, представляет собой наиболее доступное решение. В будущем планируется расширить функциональность приложения, с помощью добавления централизованного подключения к единому серверу, а также возможности формировать отчёты об успеваемости школьника и добавлять их в электронный дневник.

Приложение для тестирования школьников по основам информационной безопасности может стать началом массового распространения уроков информационной безопасности в школах на постоянной основе, а значит поможет значительно увеличить уровень компьютерной грамотности в обществе.

Разработанное приложение поможет учителю проводить занятия не только по основам информационной безопасности, но и с помощью функции добавления новых тем и тестов, учителя всех школьных предметов смогут проводить свои занятия как в компьютерном классе, так и в режиме дистанционного обучения. Следовательно, с помощью приложения для тестирования школьников, проведение контроля знаний станет гораздо проще и доступнее, значительно снизит нагрузку на учителя, поскольку уменьшит количество бумажных проверочных тестов и контрольных работ и снимет с учителя обязанность по проверке этих работ.

Литература

1. Грофф. SQL: полное руководство / Грофф, Р. Джеймс, Вайнберг, Н. Пол, Оппель, Дж. Эндрю ; 3-е изд. : Пер. с англ. – СПб. : ООО «Диалектика», 2019. – 960 с.

2. Джейсон Visual C# .NET. Полное руководство / Джейсон, Майк Прайс ; Гандэрлой. – М.: Корона Принт, 2004. - 960 с.

3. Мак-Дональд Мэтью. WPF 4: Windows Presentation Foundation в .NET 4.0 с примерами на C# 2010 для профессионалов / Мэтью Мак-Дональд ; Пер. с англ. – М. : ООО «И. Д. Вильямс», 2011. – 1024 с.

4. Партыка Т.Л. Информационная безопасность / Т.Л. Партыка, И.И. Попов. – М.: ИНФРА-М, 2002. - 368 с.

УДК 519.876.3

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И АНАЛИЗ НЕЧЕТКОЙ СЕТЕВОЙ МОДЕЛИ ПРОЕКТА

Ю. С. Зырянова

Воронежский государственный университет

Введение

Для решения задач, связанных с управлением проектами, существуют специальные пакеты, но они, как правило, имеют огромный функционал, который для большинства компаний является избыточным. С другой стороны, даже если компания имеет потребность в программном обеспечении для планирования проектов и контроля за их выполнением, то оно может оказаться крайне дорогостоящим. Таким образом, актуальными являются подходы для развития методов решения различных задач, связанных с управлением проектами, которые учитывают специфику деятельности компании и особенности доступной информации. В рамках управления проектами особое значение имеет задача календарного планирования, так как именно от качества ее решения в значительной мере зависит успех реализации проекта. Решение задачи календарного планирования позволяет определить моменты начала и окончания каждой работы, резервы времени, а также определить время, необходимое для реализации проекта (критическое время). Цель статьи заключается в представлении результатов исследования сетевой модели проекта в условиях, когда информация о продолжительности работ задана приближенно в форме нечетких чисел. Для проведения вычислительного эксперимента разработана программа, позволяющая автоматизировать расчеты временных параметров.

1. Классический алгоритм определения временных параметров сетевого графика

1.1 Основные определения

Введем основные понятия, базируясь, например, на [1].

Сеть — это бесконтурный взвешенный ориентированный граф, который является математической моделью проекта. Сеть также называется сетевым графиком.

Для отображения логической последовательности выполнения и взаимосвязи работ проекта будем использовать сетевую модель в виде сетевого графика, построенного так, что каждой работе соответствует дуга с весом, равным продолжительности работы, а каждая вершина — это событие, заключающееся в том, что работы, соответствующие входящим дугам, завершены, а работы, соответствующие выходящим дугам, готовы к выполнению.

Временные параметры сетевого графика позволяют разработать календарный план реализации проекта, отражая сроки выполнения запланированных работ, и определить время, необходимое для успешного завершения проекта.

Пусть работа над проектом начинается в момент времени, равный нулю, а длина дуги соответствует продолжительности выполнения работы, тогда временем наступления события будем считать время окончания всех работ, предшествующих данному событию. Каждому i-му событию можно поставить в соответствие следующие параметры:

$t_{erl}(i)$ – раннее время наступления i-го события – время, раньше которого событие i наступить не может, иначе не будут завершены работы, которые ему предшествуют;

$t_{lst}(i)$ – позднее время наступления i-го события – время, позже которого событие i наступить не должно, иначе увеличится время, необходимое для реализации проекта.

Заметим, что $t_{erl}(i) \le t_{lst}(i)$, при этом величина $R(i) = t_{lst}(i) - t_{erl}(i)$ называется резервом времени i-го события.

Зафиксируем событие i. Очевидно, что для его наступления необходимо, чтобы все работы, которые ему предшествуют, были завершены. Тогда раннее время $t_{erl}(i)$ можно интерпретировать как длину максимального пути из исходного события в данное i-е событие.

Длина максимального пути из исходного события в завершающее – это время, необходимое для реализации всего проекта, так как время завершения всего комплекса работ не может быть меньше, чем суммарная продолжительность всех работ вдоль самого длинного пути. Это время называется критическим и обозначается $T_{\max}(i)$. Сетевой график может иметь несколько критических путей. Все они имеют одинаковую длину $T_{\max}$.

Позднее время $t_{lst}(i)$ наступления i-го события можно интерпретировать как разность между длиной критического пути и длиной максимального пути из i-го события в завершающее событие сети.

Работы и события, принадлежащие критическому пути, также называются критическими. Особенность критических работ заключается в том, что они не имеют резервов времени, поэтому невыполнение срока окончания любой из таких работ приводит к увеличению критического времени $T_{\max}(i)$. В связи с этим именно критические работы требуют бесперебойного обеспечения ресурсами и контроля за выполнением. Все остальные работы располагают определенными резервами времени.

Имеет место следующее утверждение: для того, чтобы событие i принадлежало критическому пути необходимо и достаточно, чтобы раннее и позднее времена наступления этого события совпадали, то есть $t_{erl}(i) = t_{lst}(i)$.

1.2 Алгоритм определения временных параметров сетевого графика

1 шаг. Топологическая сортировка сети

Получить правильную нумерацию вершин сетевого графика, при этом исходное событие сети получает номер 0, а завершающее – номер n (под правильной нумерацией вершин бесконтурного графа подразумевается такая нумерация, согласно которой для каждой дуги (i, j) номер начала i меньше номера конца j).

2 шаг. Определение ранних времен наступления событий.

Положить $t_{erl}(0) = t_{erl}(1) = \ldots = t_{erl}(n)$. Двигаясь по пронумерованной сети в порядке возрастания вершин, для каждой вершины j определить

$$t_{erl}(j) = \max_{i \in \Gamma^{-1}(j)} \left\{ t_{erl}(i) + t_{ij} \right\},$$

где $\Gamma^{-1}(j) = \{ i : i \to j \}$ — множество вершин, из которых дуги ведут в вершину j.

3 шаг. Определение критического пути.

Положить $T_{\max} = t_{erl}(n)$, где n — завершающее событие сети. Для определения критического пути выполнить следующие действия: из всех дуг, входящих в завершающее событие n, выделить те дуги (i, j), которые удовлетворяют условию $t_{erl}(j) - t_{erl}(i) = t_{ij}$.

Затем рассматриваются те вершины, из которых выходят выделенные дуги, и снова из входящих в них дуг выделяются те, которые удовлетворяют тому же условию. Процесс продолжается до тех пор, пока не будет достигнуто исходное событие сети. Путь из исходного события в завершающее событие, составленный из выделенных дуг, является критическим.

4 шаг. Определение поздних времен наступления событий.

Положить $t_{lst}(n) = t_{erl}(n) = T_{\max}$. Двигаясь по сети от завершающего события в порядке убывания номеров вершин, определить для каждого i-го события

$$t_{lst}(i) = \min_{i \in \Gamma(i)} \left\{ t_{erl}(i) - t_{ij} \right\},$$

где $\Gamma(i) = \left\{ j : i \to j \right\}$ — множество вершин, в которые ведут дуги из i.

5 шаг. Определение резервов времени.

Для каждой работы (i, j) определить резервы времени $R_{full}(i, j)$, $R_{free}(i, j)$, $R_{indep}(i, j)$.

Для определения, является ли событие i критическим, можно использовать его резерв времени $R(i)$, который показывает, насколько можно задержать наступление этого события, не вызывая при этом увеличения критического времени. Для критических событий данный резерв равен нулю, для некритических – положителен.

Пусть i и j — события, соответствующие началу и окончанию некоторой работы, тогда будем обозначать ее (i, j). В сетевом графике данной работе соответствует дуга с тем же обозначением (i, j) и весом t_{ij}, равным продолжительности работы. К временным параметрам сетевого графика также относятся резервы времени работ.

Полный резерв времени работы (i, j)

$$R_{full}(i, j) = t_{lst}(j) - t_{erl}(i) - t_{ij}$$

есть разность между критическим временем выполнения проекта $T_{\max}$ и максимальной длиной пути, проходящего через эту работу, поэтому это максимальное время, которым можно располагать для увеличения продолжительности работы (i, j) без увеличения $T_{\max}$.

Свободный резерв времени работы (i, j)

$$R_{free}(i, j) = t_{erl}(j) - t_{erl}(i) - t_{ij}$$

есть максимально допустимое время увеличения продолжительности работы (i, j) при котором все работы (j, k), предшествующие событию j, начинаются в ранее время $t_{erl}(k)$.

Независимый резерв времени работы (i, j)

$$R_{indep}(i, j) = max\left\{0, t_{erl}(j) - t_{lst}(i) - t_{ij}\right\}$$

есть максимально допустимое количество времени для увеличения продолжительности работы (i, j) при условии, что все работы (k, i), входящие в i-е событие, заканчиваются в позднее время $t_{lst}(i)$, а все работы (j, m), выходящие из события j, начинаются в раннее время $t_{erl}(j)$.

Отрицательное значение независимого резерва означает, что любая задержка в выполнении работы приведет к дополнительным ограничениям на выполнение других работ.

2. Постановка задачи

В задаче рассматривается граф, в котором продолжительности работ заданы нечеткими числами, то есть работа из вершины 0 в вершину 1 задана как «приблизительно 2», из вершины 1 в вершину 2 «приблизительно 3» и т. д. На рис. 1 представлен граф с модальными значениями тройки чисел нечеткого треугольного числа (a,l,r), с функцией принадлежности

$$\mu(x)=\begin{cases}1-\dfrac{a-x}{l}, \textit{если } a-l\le x\le a,\\ 1-\dfrac{x-a}{r}, \textit{если } a\le x\le a+r,\\ 0, \textit{иначе},\end{cases}$$

где a – модальное значение (точка экстремума) функции принадлежности $\mu(x)$, l – левый коэффициент неопределенности; r – правый коэффициент неопределенности.

Таблица 1

Продолжительности работ в форме треугольных нечетких чисел

	0	1	2	3	4	5	6	7	8
0	0	(1,2,4)	0	0	0	0	0	0	0
1	0	0	(2,3,5)	(2.5,3.5,5.5)	(3,4,6)	0	0	0	0
2	0	0	0	0	0	(3,4,6)	0	0	0
3	0	0	0	0	0	0	(4,5,7)	0	0
4	0	0	0	0	0	(2.5,3.5,5.5)	0	(5,6,8)	0
5	0	0	0	0	0	0	0	(3.5,4.5,6.5)	0
6	0	0	0	0	0	0	0	(4.5,5.5,7.5)	0
7	0	0	0	0	0	0	0	0	(1.5,2.5,4.5)
8	0	0	0	0	0	0	0	0	0

Важнейшим понятием нечеткого моделирования является понятие α-среза [2], который для треугольного нечеткого числа имеет вид $(a,l,r)_\alpha=\left[a-l(1-\alpha),a+r(1-\alpha)\right]$.

Пусть исходная информация о продолжительностях работ проекта задана в табл. 1.

Сетевой график проекта представлен на рис. 1.

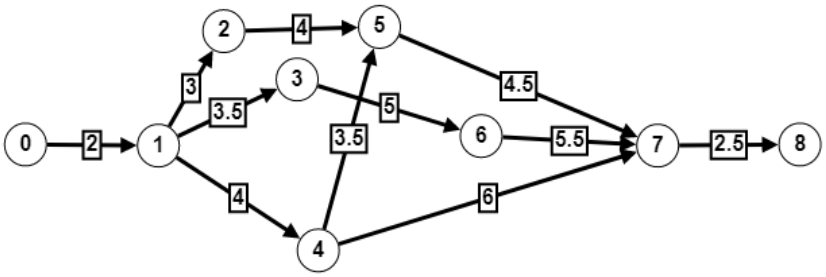


Рис.1. Сетевой график

Необходимо рассчитать критическое время реализации проекта для сетевого графика в условиях неопределённости и исследовать точность получаемого решения при различных параметрах α. Параметры l, r были взяты равными 1 и 2 соответственно. Будем исследовать результаты для α-срезов равных 0.2, 0.4, 0.7, 0.9, 1. При α=1 должно быть найдено точное решение.

На рис. 2 показан критический путь, проходящий через вершины (0-1-3-6-7-8).

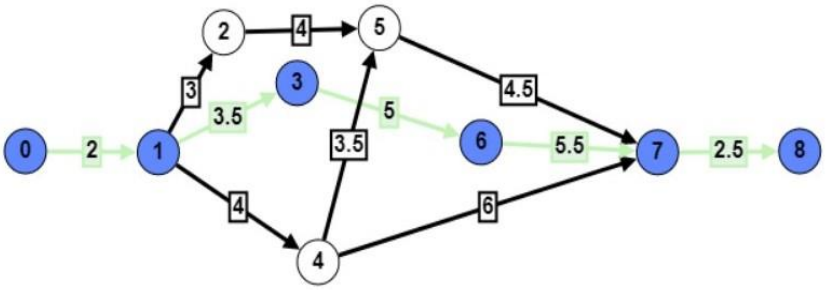


Рис. 2. Критический путь

3. Программная реализация

Для написания программы был использован язык C# с использованием интегрированной среды разработки Visual Studio. При запуске программы открывается окно, представленное на рис. 3. Сначала нужно ввести количество событий, после чего заполнить поля с параметрами $l, r, alpha$, и в появившемся окне ниже заполнить данные матрицы смежности графа, представленного на рис. 1.

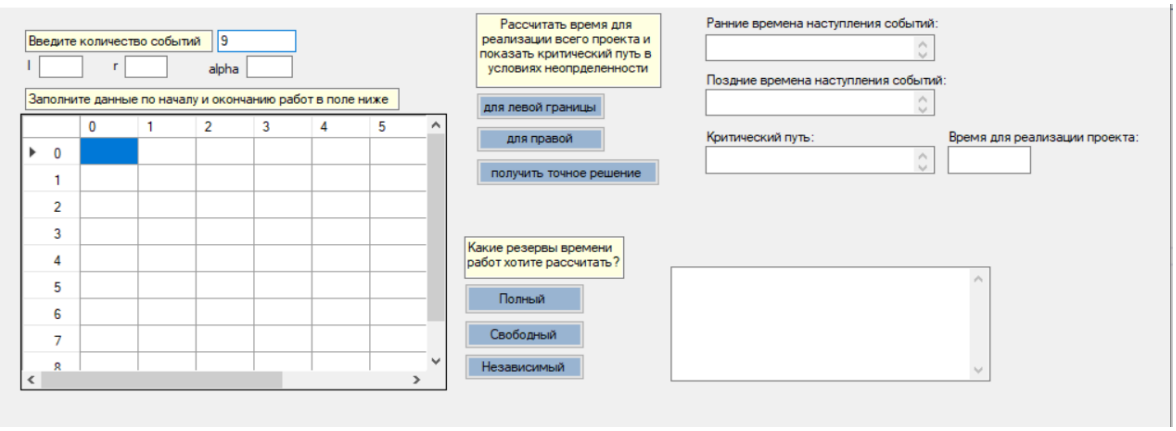


Рис. 3. Окно ввода данных

После заполнения всех необходимых данных, можно получить решение (время реализации проекта) для левого и правого значений нечеткого треугольного числа при определенных параметрах *alpha* и точное решение, нажав на советующие кнопки (рис. 4).

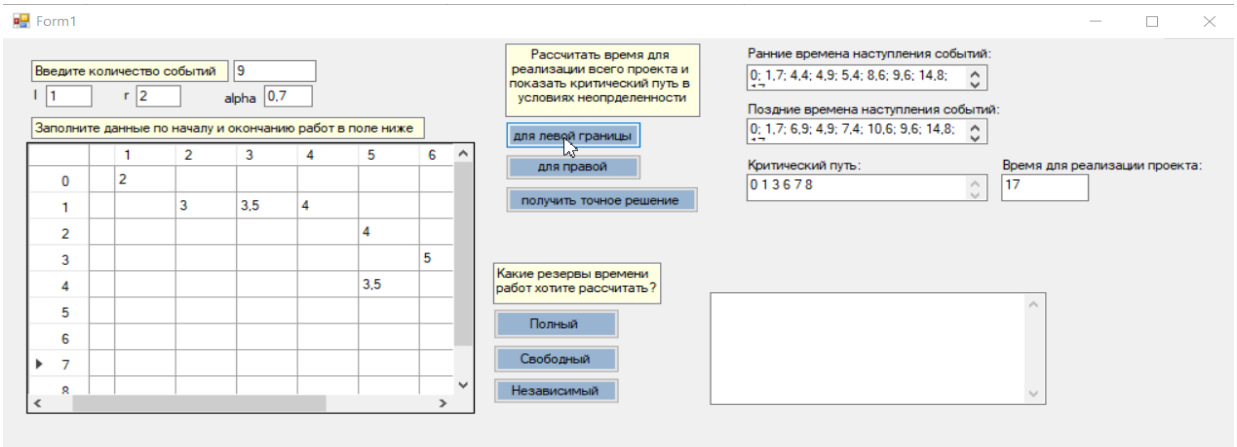


Рис. 4. Окно вывода результатов расчета критического времени T_{max} *и критического пути*

Также в программе реализованы процедуры нахождения резервов времени (рис. 5).

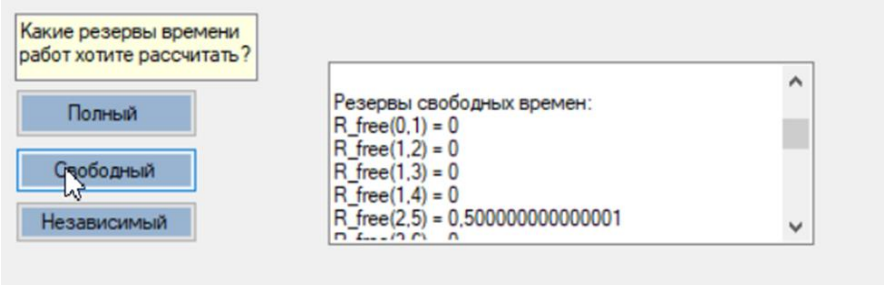


Рис. 5. Окно вывода результатов нахождения резервов времени

В результате вычислительного эксперимента для рассматриваемой задачи на основе полученных значений в ходе работы программы был построен график в Excel, представленный на рис. 6.

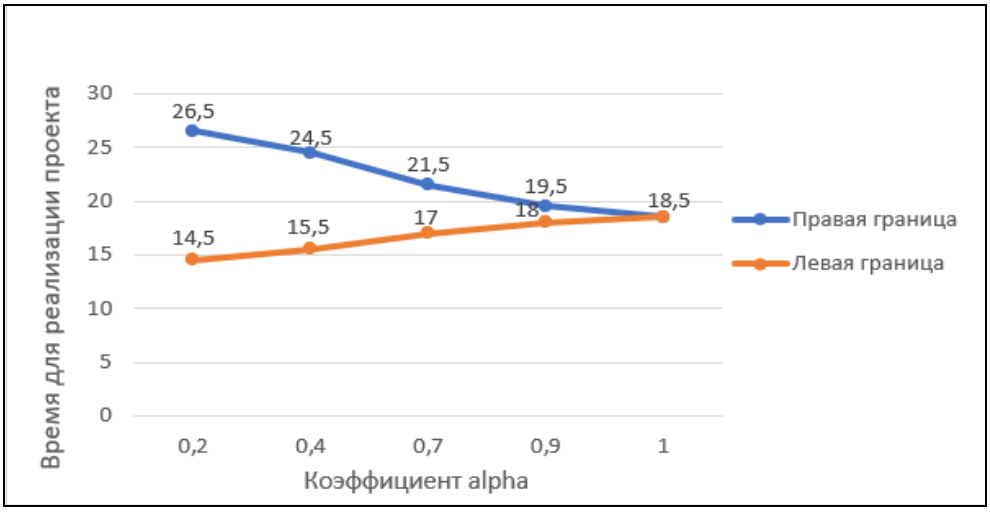


Рис. 6. График зависимости времени реализации проекта от коэффициента $alpha$

Из графика четко видно, что при стремлении параметра $alpha$ к 1 решение приближается к точному решению с модальными значениями, а при $alpha = 1$ достигается точное решение.

Заключение

В рамках проведенного исследования был изучен классический алгоритм календарного планирования. В предположении, что продолжительности работ заданы нечеткими треугольными числами рассмотрена модификация сетевой модели, которая основана на использовании формулы декомпозиции, что позволяет для заданного значения параметра α получить задачу с интервальными продолжительностями работ на основе α-срезов. Был проведен анализ сетевой модели проекта и изучена зависимость точности решения от параметра α. В дальнейшем планируется использовать в алгоритме нахождения критического пути нечеткие операции $\max$ и $\min$, а также другие типы нечетких чисел.

Литература

1. Леденев М.Ю., Сергиенко М.А. Алгоритм расчета нечетких и интервальных оценок временных параметров сетевой модели проекта [Электронный ресурс] / Леденев М.Ю., Сергиенко М.А.// Международный научно-исследовательский журнал – 2021. – № 6. – Режим доступа: https://research-journal.org/wp-content/uploads/2021/06/6-108-1.pdf#page=118. – (Дата обращения: 18.04.2023).

2. Леденева Т.М. Обработка нечеткой информации : учебное пособие / Т.М. Леденева. – Воронеж : Воронежский государственный университет, 2006. – 233 с.

УДК 004

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ НОВОСТНОГО ПОРТАЛА ДЛЯ ФАКУЛЬТЕТА

В. А. Иванников

Воронежский государственный университет

Введение

Со временем, технологии, разработки сайтов устаревают. Для реализации новой функциональности программистами тратится больше времени, чем для сайтов с более современным стеком технологий; устаревшие сайты работают медленнее в сравнении с более современными. С момента создания некоторых новостных сайтов, разработанных для факультета, прошло много времени. В связи с этим возникла потребность разработать новый вариант новостного портала с возможностью интеграции с существующими веб-страницами новостного портала для факультета. В работе рассматривается реализация серверной части новостного портала.

1. Функциональность приложения и его реализация

Типовой сайт должен поддерживать следующую функциональность:

1. Просмотр добавленных статей.
2. Регистрация и аутентификация пользователей.
3. Шифрования пароля на основе криптографической хеш-функции для авторизации пользователей.
4. Добавления статьи на сайт.
5. Поиск статьи на основе алгоритма хеширования.
6. Процедура редактирования статьи.
7. Процедуры удаления статьи.

Для реализации серверной части были использованы следующие инструменты:
Фреймворк на сервере: Spring Boot 3
Инструмент сборки: Maven 3
СУБД: PostgreSQL
Миграции БД: Liquibase
Способ авторизации: Spring Security+JWT
Библиотека тестирования: JUnit5
ПО для контейнеризации: Docker

Программный код представляет из себя RESTful-сервис для новостного портала. Все приложение разделено на несколько «программных слоев»: Controller-layer, Service-layer, Repository-layer. При реализации компонентов были применены принципы «слоеной архитектуры» [1], принципы SOLID, REST.

При отправке запроса из браузера, запрос обрабатывается перехватчиком запросов на сервере. После этого запрос попадает в фильтры, где сервер авторизует запрос на выполнение действий. Если запрос был авторизован сервером, то запрос попадает на диспетчер сервлетов, где запрос перенаправляется на нужный контроллер. Если запрос не был авторизован, то в браузер отправляется сообщение «Ошибка доступа». Контроллер имеет возможность принимать запрос от клиента и отправлять пользователю ответ. Он отправляет информацию

(DTO / параметры запроса), которая пришла из браузера на слой сервисов. Сервисы выполняют бизнес-логику приложения и обращаются к репозиторному слою. Репозиторий предназначен для работы с базой данных (рис. 1).

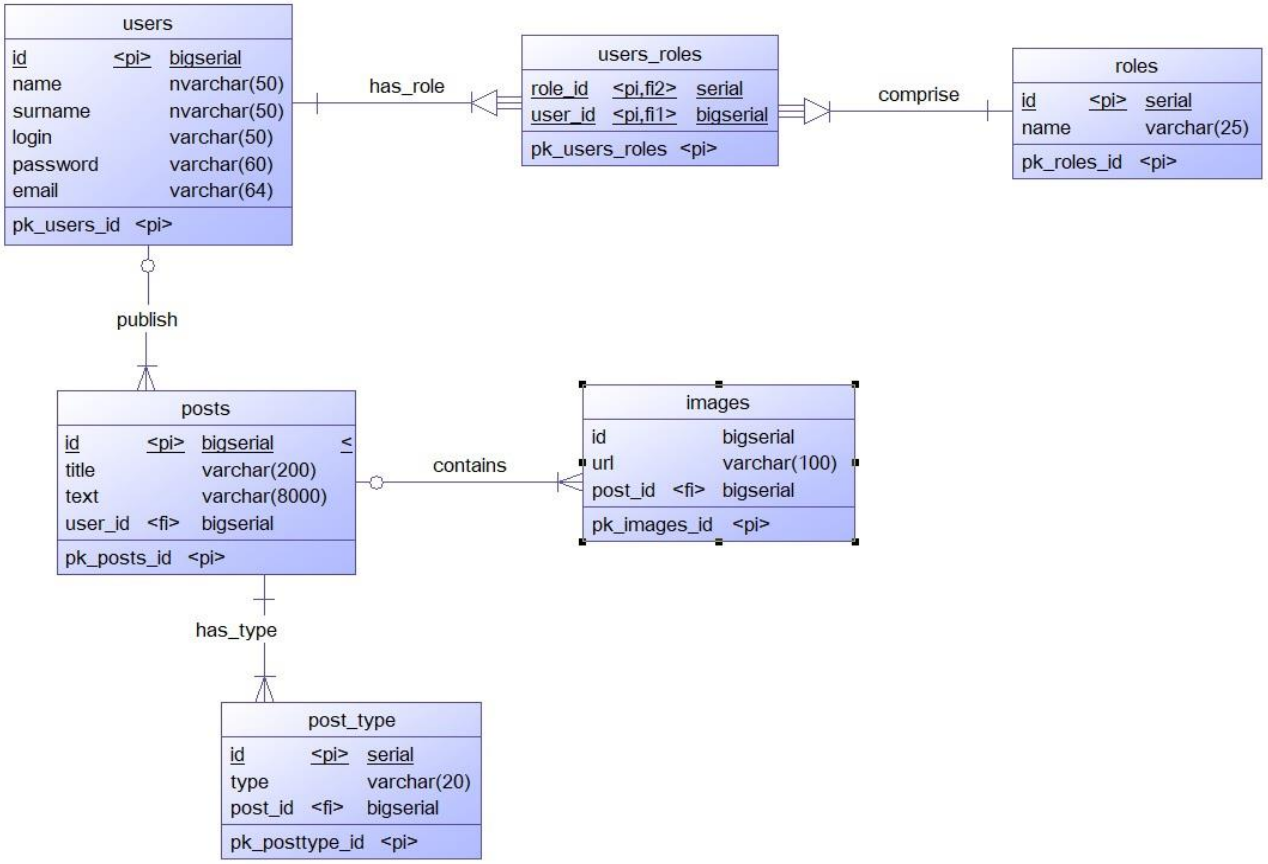


Рис. 1. Логическая модель БД

База данных состоит из нескольких сущностей. Сущность users содержит информацию обо всех зарегистрированных пользователях. Сущность roles содержит информацию о ролях пользователей в системе. Сущность users_roles является «промежуточной сущностью» между сущностями users и roles. Сущность posts предназначена для хранения информации о постах. Сущность post_type содержит информацию о возможных типах поста. Сущность images предназначена для хранения уникальных названий картинок.

Для обработки ошибок, которые могут происходить в сервисе, был реализован обработчик ошибок, который выдает сообщения об исключениях.

Для авторизации пользователя используется библиотека JWT [2]. JWT-токен содержит информацию об имени пользователя и его правах доступа. Внутри jwt-токена содержится информация об имени пользователя и его правах доступа. Когда пользователь отправляет запрос, он перехватывается и обрабатывается фильтрами. Фильтры получают из заголовков запроса авторизационный токен и проверяют имя пользователя и права доступа к какому-либо ресурсу. Авторизационный токен присутствует в запросах на добавление, удаление, редактирование статей и загрузке картинок. Он не содержится в запросах на получение статьи по идентификатору, на получение списка статей, регистрации, аутентификации, поскольку эти операции пользователь может выполнить не будучи авторизированным. Последовательность действий, которые выполняются сервисом при попадании запроса на сервер, представлены на рис. 2.

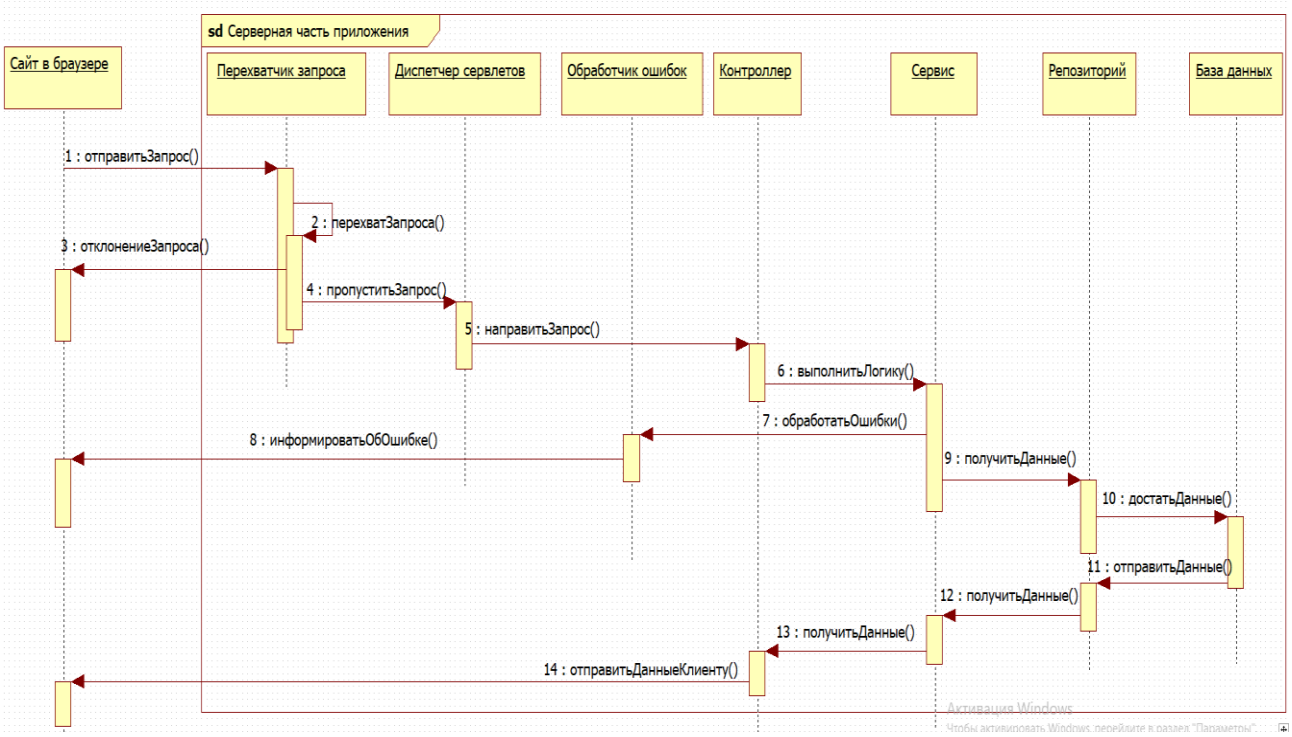


Рис. 2. Последовательность действий при попадании запроса на сервер

2. Дополнительные возможности

Для разворачивания приложения на сервере, API был собран в docker-контейнере. Этот способ развертывания приложения гарантирует запуск сервиса на любом сервере вне зависимости от операционной системы.

Для тестирования приложения была использована библиотека JUnit5. Модульные тесты проверяют функционирование сервиса и повышают скорость, безопасность написания кода при реструктуризации приложения.

Для тестирования приложения в GitHub и отправления обновленного docker-контейнера в DockerHub был настроен инструмент GitHub Actions.

Для возможности миграции с одной СУБД на другую, база данных была написана на скриптовом языке Liquibase.

Заключение

Использование современного стека технологий при реализации серверной части обеспечит быстрый ввод новой функциональности (при необходимости), увеличится скорость обработки запроса. А также, стек технологий, используемый при разработке новостного портала, останется актуальным еще долгое время.

Литература

1. Гексагональная архитектура. – Режим доступа: https://habr.com/ru/articles/267125/. – (Дата обращения: 14.04.2023)

2. Introduction to JSON Web Tokens. – Режим доступа: https://jwt.io/introduction. – (Дата обращения: 08.04.2023)

УДК 004.432.2

ИСПОЛЬЗОВАНИЕ REACT И NODE.JS ДЛЯ РАЗРАБОТКИ

А. И. Камышанов

Воронежский государственный университет

Введение

В современном мире создание полноценного интернет-магазина является важной задачей для многих компаний и предпринимателей. Для ее выполнения необходимо использо- вать современные технологии и фреймворки. В данной статье рассмотрено применение двух таких технологий - React и Node.js - для создания полноценного многопользовательского интернет-магазина. Рассмотрим основные принципы работы React и Node.js и их использование в контексте разработки интернет-магазина. Также будут рассмотрены вопросы, связанные с проектированием и разработкой архитектуры приложения, взаимодействием клиентской и серверной частей, работой с базой данных и обеспечением безопасности приложения. В итоге будет получен полноценный проект, готовый к запуску в продакшн и обеспечивающий комфортную и безопасную работу для многих пользователей.

Node.js для серверной части

Для серверной части мы будем использовать Node.js - среду выполнения JavaScript на серверной стороне, которая позволяет исполнять JavaScript код на сервере. В контексте интернет-магазина Node.js используется для обработки серверных запросов и выполнения бизнес-логики приложения. Node.js разбивает приложение на несколько модулей, позволяющих отслеживать состояние на серверной стороне. Например, модуль обработки платежей может обрабатывать информацию о платежах, а модуль управления заказами - управлять информацией о заказах.

Node.js также позволяет использовать базы данных, такие как Postgres, для хранения информации о продуктах, пользователях, заказах и т.д. Это позволяет повысить производительность и управление данными приложения. Node.js также имеет хорошую поддержку для разработки API и взаимодействия с другими сервисами веб-приложения. Например, приложение может использовать API сторонних сервисов для оплаты, получения информации о доставке, отслеживания заказов и т.д. Основным преимуществом Node.js является его быстрота и производительность. Node.js работает на основе однопоточной модели, что позволяет обрабатывать множество запросов одновременно без замедления работы приложения. Также, Node.js предлагает удобный веб-сервер синтаксис, который делает кодирование на нем быстрым и простым. В целом, Node.js является мощным инструментом для разработки серверных приложений, в том числе для интернет-магазинов, благодаря своей производительности, удобству разработки и возможностям взаимодействия с другими сервисами.

React для пользовательской части

React - это библиотека JavaScript, которая используется для разработки пользовательских интерфейсов. Она позволяет создавать динамические и масштабируемые интерфейсы, которые

реагируют на действия пользователя. В контексте интернет-магазина, React используется для создания интерфейса для пользователей, включая страницы продуктов, корзины, оформления заказа и т.д. React упрощает создание компонентов пользовательского интерфейса, которые могут использоваться повторно в разных частях приложения. React также может оптимизировать производительность приложения и минимизировать обновления страницы при изменении данных, используя виртуальную DOM. Важно отметить, что в сфере интернет-магазинов время отклика приложения играет ключевую роль, и React позволяет достичь быстродействия при работе с большими объемами данных. React также может работать с другими библиотеками и фреймворками, такими как Redux или MobX, для управления состоянием приложения и улучшения работы с данными.

Работа с данными

Была использована PostgreSQL - мощная система управления базами данных, предоставляющая множество функций и возможностей для работы с данными, таких как работа со строковыми, числовыми, датами/временем, JSON и другими типами данных. Также была использована ORM Sequelize, которая автоматически создает SQL-запросы на основе объектов моделей данных, упрощая работу с базой данных и повышая производительность. Sequelize предоставляет возможность создания сложных запросов на основе SQL, а также работы с транзакциями и агрегатными функциями.

Использование технологий вместе

Был создан сервер Node.js, который возможно использовать для обработки запросов от клиента и обращения к базе данных PostgreSQL через Sequelize. База данных PostgreSQL с её таблицами следовала за сервером, чтобы хранить данные приложения. Модели Sequelize были созданы соответственно таблицам из базы данных, и взаимосвязи между ними были определены. На очереди было создание API-маршрутов Node.js для обработки запросов от клиента. Для этого маршруты использовали модели Sequelize для взаимодействия с базой данных. В конце, клиентская часть приложения была создана с помощью React, что позволило клиенту взаимодействовать с сервером посредством API-маршрутов. Функциональность веб-приложения на серверной и клиентской сторонах была реализована с помощью JavaScript и соответствующих библиотек (рис.1).

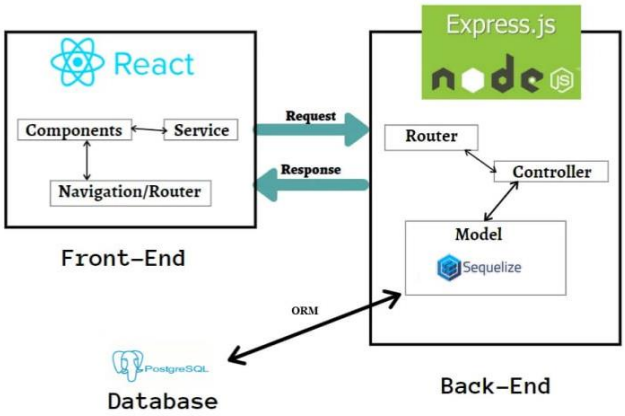


Рис. 1. Схема работы приложения

Заключение

В заключении можно отметить, что использование Node.js для серверной части и React для пользовательской части является эффективным решением для разработки интернет-магазина. Node.js позволяет быстро и эффективно обрабатывать запросы на сервере, уп- равлять базой данных и взаимодействовать с другими сервисами, а React обеспечивает удобную и оптимизированную работу с пользовательским интерфейсом, что особенно важно для интернет-магазинов. Использование этих инструментов позволяет создать масштабируемое и быстродействующее приложение с удобным и функциональным интерфейсом для пользователей.

Литература

1. Фриман, Э. Реакция: Up & Running: Создание веб-приложений. – Себастопол, Калифорния: O’Reilly Media, 2015. – 348 с.
2. Кантелон, Г. Node.js для веб-разработчиков, 2-е издание: Веб-разработка на стороне сервера упрощается с помощью node.js. – Себастопол, Калифорния: O’Reilly Media, 2012. – 192 с.
3. Хьюз, П. Pro MERN Stack: разработка веб-приложений полного стека с помощью Mongo, Express, React и Node. – Беркли, Калифорния: Апресс, 2019. – 805 с.
4. Уилсон, М. Проекты React с полным стеком: Изучите разработку MERN stack, создавая современные веб-приложения с использованием MongoDB, Express, React и Node.js . – Бирмингем, Великобритания: Packt Publishing, 2018. – 328 с.

УДК 004.7

ИССЛЕДОВАНИЕ ВРЕМЕНИ ПЕРЕДАЧИ ПАКЕТОВ В КОМПЬЮТЕРНЫХ СЕТЯХ

И.С. Кириллов

Воронежский государственный университет

Введение

Анализ развития информационных систем показывает резкое увеличение количества беспроводных устройств, использующих Wi-Fi. Вместе с этим возрастает загруженность сети [1], что может значительно ухудшать качество функционирования используемых беспроводных устройств. Поэтому при моделировании, оценке эффективности функционирования сети, диагностике устройств часто возникает необходимость в оценке загруженности сети, которую можно оценить скоростью передачи, временем передачи пакетов для пакетов реального времени, количеством потерянных пакетов.

Наиболее информативным является время передачи пакетов, т.к. среднее время передачи пакетов заданного размера позволяет определить скорость передачи, а время передачи, превышающее заданное позволяет определить потерянные пакеты.

Поэтому разработка алгоритмического и программного обеспечения для экспериментального определения времени передачи между устройствами сети и исследования результатов актуальна.

1. Разработка ПО и анализ

Целью данного исследования является разработка программного обеспечения для экспериментального определения времени передачи пакетов и анализа результатов. Для этого необходимо решить следующие задачи:

- разработка ПО для отправки и получения пакетов,
- разработка методики экспериментального исследования,
- разработка ПО для анализа результатов,
- анализ результатов экспериментов.

При моделировании процесса передачи пакетов по сети наиболее частот используют экспоненциальное распределение, поэтому в работе использовано указанное распределение.

1.1. Разработка ПО для отправки и получения пакетов

При разработке ПО используется архитектура «клиент–сервер». Принцип работы ПО разделяется на две части: программа-клиент и программа-сервер. Вначале запускается программа-сервер. В ней создается и настраивается для приема сокет, устанавливаются параметры закона распределения. Программа-сервер переходит в режим ожидания. Запускается программа-клиент. В ней создается массив времени отправки пакета по закону распределения с помощью заголовочного файла random [4] и записывается в отдельный файл. После этого создается и настраивается для отправки сокет. Отправляется первый пакет и срабатывает команда задержки, которая реализована в заголовочном файле chrono [5]. В этом момент программа-сервер начинает принимать пакеты и записывать время получения каждого в массив. Затем по очереди отправляются все остальные пакеты с использованием задержки

между ними. После передачи всех пакетов отправляется файл с массивом времени, сокет закрывается, и программа-клиент завершает свою работу. После получения последнего пакета и файла программа-сервер закрывает сокет и формирует файл, в который записывается массив времени получения пакетов. На этом завершается цикл обмена данными и запускается программа для анализа. Программа-сервер открывает сокет и переходит в режим ожидания.

Принцип обмена информации между двумя программами (рис. 1):

1. Создание массива времени отправки пакета по закону распределения.
2. Открытие сокета для передачи данных
3. Отправка пакетов по времени, указанному в массиве
4. Закрытие сокета
5. Обработка полученных данных

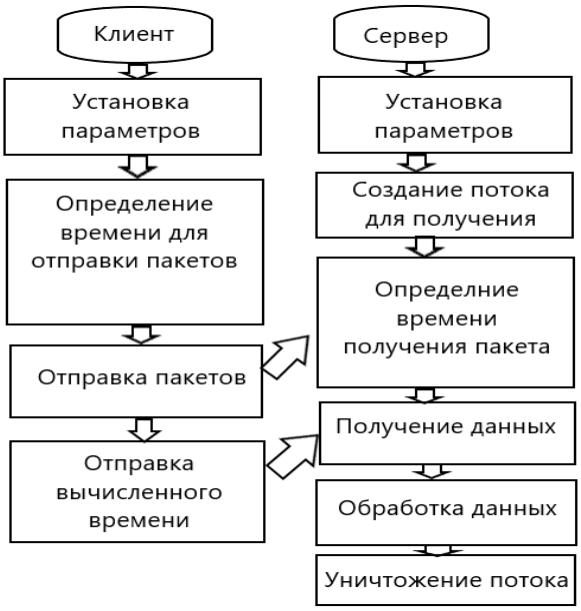


Рис. 1 Схема обмена информации

1.2. Разработка методики экспериментального исследования

Для проведения исследования времени отправки и принятия пакета была произведена передача данных в количестве 1000 пакетов при разной загрузке компьютерной сети.

При этом были приняты следующие допущения:

• время прохождения пакета и время подтверждения считаем одним событием, так как они зависят от загрузки канала и интенсивности отправки передаваемых пакетов

• не учитывается установление соединения, так как оно не влияет на время передачи пакетов потому, что используется протокол UPD [3], который не использует предварительное сообщение для установки специальных каналов передачи.

Была составлена схема устройства сети (рис. 2), на которой слева представлены две ЭВМ использующие разработанное ПО. При помощи беспроводной локальной сети и Wi-Fi роутера ЭВМ обмениваются информацией.

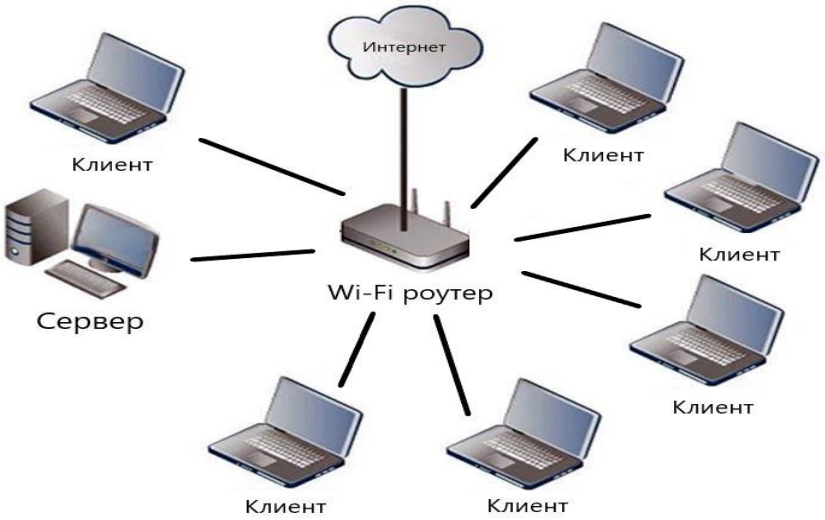


Рис. 2 Схема устройства сети в эксперименте

1.3. Разработка ПО для анализа результатов

Для анализа результатов было разработано отдельное ПО, которое использует файлы, полученные при отправки и получении пакетов. Принцип работы ПО:

1. Считывание файлов и запись данных в массивы
2. Выполняются различные вычислительные преобразования с данными
3. Выводится несколько графиков на экран при помощи библиотеки matplotlib [6].

Результаты проведенного исследования (рис. 3) при соединении двух объектов сети и передачи информации с использованием формулы:

$$f(x) = \begin{cases} 0, & x < 0 \\ \lambda e^{-\lambda x}, & x \geq 0 \end{cases}$$

где λ – интенсивность поступления пакетов для передачи.

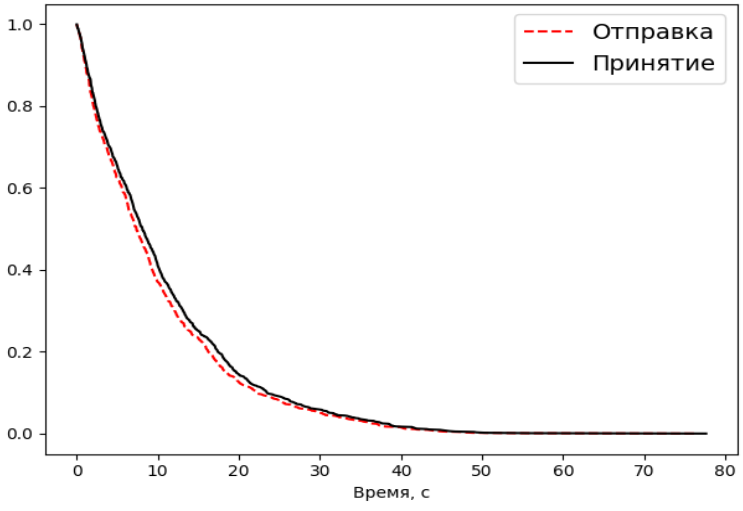


Рис. 3 График плотности распределения

1.4. Анализ результатов экспериментов

Далее были произведены эксперименты при разной загруженности сети. Для этого тестирования изменялось количество подключенных и работающих в сети клиентов без учета ЭВМ, выполняющих разработанное ПО. Полученные из эксперимента данные сравниваются на рис. 4, в котором посчитана относительная разница между вычисленным временем отправки пакета и временем при получении пакета в зависимости от количества клиентов в сети.

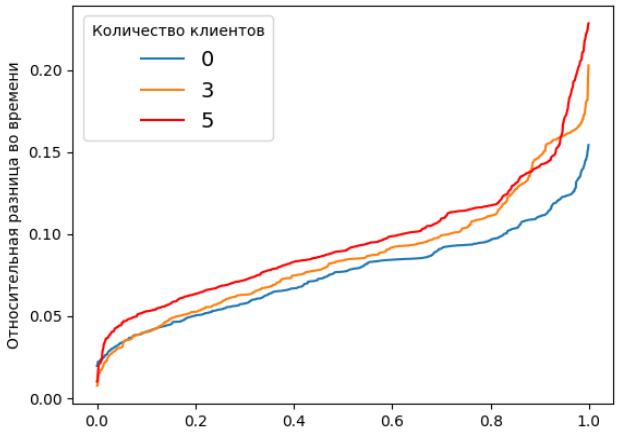


Рис. 4 График относительной разницы во времени при разной загруженности

Данные результаты иллюстрируют закономерность изменения времени от загруженности сети: чем выше загруженность, тем больше максимальная относительная разница времени передачи пакетов.

Заключение

Таким образом, было разработано ПО, с помощью которого можно адекватно оценить время доставки для систем с заданными характеристиками.

Результаты исследований могут быть применены при создании систем реального времени с протоколами UPD и настройке параметров их функционирования.

Литература

1. Ампелонский, А. Ю. Рост уровня помех в работе беспроводных сетей, обусловленный увеличением числа сетей / А. Ю. Ампелонский. — Текст : непосредственный // Молодой ученый. — 2019. — № 46 (284). — С. 9-11.
2. Гмурман, В. Е. Руководство к решению задач по теории вероятностей и математической статистике: учебное пособие для студентов вузов / В. Е. Гмурман. − М.: Высш. школа, 2004. − 404 с.
3. Стивенс У. Р., Феннер Б., Рудофф Э. М. UNIX: разработка сетевых приложений / Стивенс У. Р., Феннер Б., Рудофф Э. М. – 3-е изд., перераб. и доп. – СПб : Питер, 2007. – 1039 с.
4. Microsoft. Documentation. C++ Standard Library header files. <random>. – Режим доступа: https://learn.microsoft.com/en-gb/cpp/standard-library/random?view=msvc-160 (Дата обращения: 11.04.2023).
5. Microsoft. Documentation. C++ Standard Library header files. <chrono>. – Режим доступа: https://learn.microsoft.com/en-gb/cpp/standard-library/chrono?view=msvc-160 (Дата обращения: 11.04.2023).
6. Matplotlib: Visualization with Python. – Режим доступа: https://matplotlib.org/ (Дата обращения: 12.04.2023).

УДК 519.874

УПРАВЛЕНИЕ МНОГОПРОДУКТОВЫМИ ЗАПАСАМИ ПРИ КОМБИНИРОВАННЫХ ПОСТАВКАХ.

Е.А.Коваленко

Воронежский государственный университет

Введение

Одна из наиболее важных составляющих успешного ведения любого бизнеса это грамотное и высокоэффективное управление запасами. Эффективное управление запасами продукции широкой номенклатуры представляет внушительную сложность для крупных компаний и предприятий, потому что для этого процесса необходимо учитывать очень много нюансов, таких как ограничения на объём партий, комбинированные поставки и планирование производства. Одним из важнейших факторов является стабильность (постоянство) спроса. В рамках данной статьи будут рассмотрены основные задачи управления многопродуктовыми запасами при условии планирования комбинированных поставок при случайном спросе, а также методы решения этих задач.

Для эффективного управления большим количеством запасов необходим постоянный мониторинг наличия товаров на складе предприятия, анализ потребности этих товаров в производстве и продаже. Управление запасами включает в себя учет товаров, находящихся на складе, и их потребностей в производстве или продаже. Компании, занимающиеся производством и реализацией многопродуктов, сталкиваются с проблемой управления запасами каждого продукта в отдельности, что делает этот процесс еще более сложным. Планирование производства и ограничения на объем партий прибавляют еще один уровень сложности в управление запасами. Вместе с тем, введение комбинированных поставок позволит облегчить управление запасами.

Складирование большого количества продукции это одна из наиболее важных, но при этом проблематичных задач. Причины проблем кроются, во-первых, в том, что большинство известных методов оптимизации данного процесса не в состоянии надежно и точно выделить высокоэффективные параметры управления для сразу всех стандартных ограничений, а также крупных партий поставок. А во-вторых, проблемой являются высокие затраты на хранение и перевозку продукции.

1. Построение систем управления многопродуктовыми запасами при случайном спросе.

Решение проблемы оптимизации управления запасами на предварительном этапе требует привлечени модели ABC-анализа. ABC-анализа представляет собой метод классификации ресурсов, при котором анализ производится на основе нескольких параметров, таких как оборачиваемость, товарный запас, вес, цена, объём и т.д. После проведения ABC-анализа ресурсы разделяются на три категории в соответствии с их важностью.

1) Категория A, которая составляет около 10-20% всех ресурсов, содержит самые ценные ресурсы компании, необходимые для ее успешной деятельности. Управление этой категорией требует качественного планирования, анализа и постоянного учета, возможно, даже ежедневного. Эта группа ресурсов является основополагающей для бизнеса компании и составляет около 75%-80% ее капитала продаж.

2) Ресурсы, относящиеся к категории B, составляют 20-40% от общих ресурсов. Они менее значимы для компании, но также требуют регулярного контроля и постоянного учета. Анализ обычно проводится ежемесячно. На их долю приходится 15-20% капитала продаж.

3) Ресурсы, относящиеся к категории C, составляют 40-70%, они представляют собой малоценные ресурсы в широком спектре, для которых достаточно использования упрощенных методов планирования, управления и учёта. Эта категория ресурсов составляет от 5 до 15% оборотного капитала компании.

Рис. 1 Процентное соотношение оборотного капитала компании по категориям ресурсов

Рассмотрим общие затраты (*D*), которые связаны с хранением, доставкой и дефицитом продукции за время *T*. Модель общих затрат будет выглядеть как:

$$D = \left(\sum_{i=1}^{L} \frac{C_{1i} \cdot s_i \cdot t_{1i}}{2} + \sum_{i=1}^{L} \frac{C_{2i} \cdot (q_i - s_i) \cdot t_{2i}}{2} + C_s \right) \cdot n \quad , \qquad (1)$$

где D – общие затраты, L — количество различных типов продуктов в группе, s_i — величина заказываемой партии продукта *i*-го типа, C_{1i} — стоимость хранения одного изделия *i*-го типа в единицу времени, C_{2i} — штраф за отсутствие *i*-го продукта на складе в течение единицы времени, q_i — спрос на продукты *i*-го типа в течение интервала времени t_s ($t_s = t_{1i} + t_{2i}$), t_s — период между закупками, t_{1i} — время, в течение которого на складе имеются запасы продукта i -го типа, t_{2i} — время, в течение которого на складе имеется дефицит продукта *i*-го типа, а $n = T/t_s$, $t_s = t_{1i} + t_{2i}$, $q_i = r_i \cdot t_s$, $s_i = r_i \cdot t_1$ (где r_i — спрос на *i*-й продукт). Соответственно, если подставим выражения для переменных n, q_i, s_i, t_{2i} в данное соотношение, то получим следующую функцию:

$$D(t_s, t_{1i}) = \left(\sum_{i=1}^{L} \frac{C_{1i} \cdot r_i \cdot t_{1i}^2}{2} + \sum_{i=1}^{L} \frac{C_{2i} \cdot r_i \cdot (t_s - t_{1i})^2}{2} + C_s \right) \cdot \frac{T}{t_s} \qquad (2)$$

Функция $D(t_s, t_{1i})$ — непрерывная функциия для переменных t_s и t_{1i} , и величина этой функции стремится к ∞ при $t_s \to 0$. Следовательно, минимум затрат на хранение продукции и

транспортировку достигается тогда, когда частная производная $\partial D(t_s,t_{1i})/\partial t_s = 0$, а частные производные $\partial D(t_s,t_{1i})/\partial t_{1i} \geq 0$ [1].

На следующем этапе проведем анализ системы контроля многопродуктовых запасов в случае постоянного спроса.

$$t_{1i} = \frac{C_{2i}}{C_{1i} + C_{2i}} \cdot t_s \tag{3}$$

Подставив выражение для t_{1i} в полученное после производной функции затрат $D(t_s, t_{1i})$ по переменной t_s соотношение, мы получим:

$$t_s^2 \cdot \left[\sum_{i=1}^{L_B} r_i \cdot \frac{C_{1i} \cdot C_{2i}}{C_{1i} + C_{2i}} + \sum_{j=1}^{L_A} r_j \cdot C_{1j}\right] - 2 \cdot C_s = 0, \tag{4}$$

где L_A — количество наименований продуктов группы A, L_B — количество наименований продуктов группы B, t_s — период между закупками.

Тогда:

$$t_s = \sqrt{\frac{2 \cdot C_s}{\sum_{i=1}^{L_B} r_i \cdot \frac{C_{1i} \cdot C_{2i}}{C_{1i} + C_{2i}} + \sum_{j=1}^{L_A} r_j \cdot C_{1j}}} \tag{5}$$

После расчета t_s и t_{1i} можно определить объем поставок $q_i = r_i \cdot t_s$ (для продуктов группы A) и соответственно $s_i = r_i \cdot t_{1i}$ (для продуктов группы B). При этом размер дефицита для продуктов группы B составит $q_i - s_i = r_i \cdot t_{2i}$.

Для обеспечения бесперебойной работы склады и компании выстраивают свою деятельность таким образом, чтобы пополнение запасов происходило периодически и в больших объемах, но с оптимальным интервалом между закупками, учитывая ограниченность ресурсов для разовых закупок и ограничения на количество товаров, которые могут быть обработаны складом за одну поставку. В связи с этим, предпочтительнее пополнять запасы партиями или группами товаров, имеющими приемлемые объемы и стоимость. Такой подход позволяет избежать излишних расходов на доставку и обеспечить оптимальное использование ресурсов.

Чтобы удовлетворить вышеуказанные требования, ассортимент товаров на складе разделяют на партии, которые необходимо регулярно пополнять, не превышая стоимость и размер каждой партии, а также принимая во внимание ограничения на доступные ресурсы и объем работ [1].

Для формализации следующей модели, предположим, что время доставки — это не случайная величина, а детерминированная. Таким образом мы будем рассматривать многопродуктовую модель управления запасами при случайном спросе [3].

Допустим, что спрос на продукцию за фиксированный период времени *T* остается стабильным, а усреднённое значение спроса на *i*-й продукт равен $\overline{r_i}$. Соответственно для определения интервала между закупками t_s будем использовать среднее значение спроса $\overline{r_i}$ в течение всего планируемого периода *T*.

$$t_s^* = \sqrt{\frac{C_s \cdot 2}{\sum_{L=1}^{n} C_i \cdot \bar{r}_i}}, \quad (6)$$

где $\bar{r}_i$ — среднее значение спроса по *i*–му продукту в течение интервала планирования *T*, C_i — стоимость хранения единицы продукта *i*-го типа в единицу времени.

Для сокращения количество недостающих или избыточных товаров на складе к моменту следующей поставки можно применить механизм обратной связи. В этом случае величина объемов закупок q_k в *k*-м периоде пополнения запасов, может быть рассчитана как:

$$q_k = \hat{r}_k t_s^* = \hat{r}_k \sqrt{\frac{2C_s}{C_1 \bar{r}}}, \quad (7)$$

где $\hat{r}_k$ — величина среднего значения спроса на *k*-м периоде пополнения запасов, а $\bar{r}$ — величина среднего значения спроса на всем интервале планирования, *q*— объем хранящейся продукции [2].

Если учитывать случайный характер спроса, то для определения интервала между закупками и размера каждой партии необходимо использовать соотношения, которые учитывают случайные величины. На основе этих соотношений будут определяться величины t_s и t_{1i}:

$$t_s = \sqrt{\frac{2 \cdot C_s}{\sum_{i=1}^{n} (\bar{r}_i \cdot \frac{C_{1i} \cdot C_{2i}}{C_{1i} + C_{2i}})}}, \quad t_{1i} = \frac{C_{2i}}{C_{1i} + C_{2i}} \sqrt{\frac{2 \cdot C_s}{\sum_{i=1}^{n} (\bar{r}_i \cdot \frac{C_{1i} \cdot C_{2i}}{C_{1i} + C_{2i}})}}, \quad (8)$$

где $\bar{r}_i$ — среднее значение спроса по *i*–му продукту на период планирования, t_{1i} — интервал времени в начале каждого периода, при отсутствии дефицита на складе продукта *i*-го типа.

Для каждого периода пополнения запасов *k* и каждого из *L* типов продуктов в группе *i* величины пополнения запасов s_{ik} ($i = 1, \ldots, L$):

$$s_{ik} = \bar{r}_{ik} t_{1i} = \bar{r}_{ik} \frac{C_{1i} \cdot C_{2i}}{C_{1i} + C_{2i}} \sqrt{\frac{2 \cdot C_s}{\sum_{i=1}^{n} (\bar{r}_i \cdot \frac{C_{1i} \cdot C_{2i}}{C_{1i} + C_{2i}})}}. \quad (9)$$

Далее остановимся на ситуации управления товарами двух групп, А и В, в соответствии с описанной выше стратегией управления многопродуктовыми запасами с постоянным спросом, но в предположении, что спрос имеет случайных характер. Для каждого вида продукта группы В время между смежными пополнениями склада определяется согласно формуле (5), после чего по формуле (3) рассчитываются величины t_{1i}. Заключительным этапом формируются объемы партий поставок для продуктов групп А и В соответственно: $q_{i=} r_i \cdot t_s$ и $S_{i=} r_i \cdot t_{1i}$.

Так как спрос у нас не постоянный, а случайный, то величины t_s и t_{1i} будут представлены соотношениями:

$$t_s = \sqrt{\frac{2 \cdot C_s}{\sum_{i \epsilon L_B} \bar{r}_i \cdot \frac{C_{1i} \cdot C_{2i}}{C_{1i} + C_{2i}} + \sum_{j \epsilon L_A} \bar{r}_j \cdot C_{1j}}}, \quad (10)$$

$$t_{1i} = \frac{C_{2i}}{C_{1i} + C_{2i}} \cdot t_s, \quad i \in L_B$$

В данной ситуации для определения размеров поставок продуктов группы A и B с учетом случайного спроса, используются соотношения: $q_{jk} = \hat{r}_{jk} \cdot t_s, j \in L_A$ для продуктов группы A с индексом *j* из множества L_A, и $S_{ik} = \hat{r}_{ik} \cdot t_{1i}, i \in L_B$ для продуктов группы B с индексом *i* из множества L_B, где $\hat{r}_{jk}$ и $\hat{r}_{ik}$ - это среднее значение спроса для продуктов группы A и B в соответствующих периодах пополнения запасов, а t_s и t_{1i} - это время между смежными пополнениями склада продукцией для каждой группы продуктов, определенное с помощью соотношений, описанных ранее.

Заключение

Таким образом, в исследовании представлена система управления запасами товаров с учетом случайного спроса и возможностью комбинировать поставки товаров разных групп. Определены расчетные характеристики для оптимального размера поставок с учетом среднего значения спроса и времени между пополнениями запасов. Отмечено, что за счет использования «обратной связи» возможно уменьшение объемов товара на складе, что должно снизить затраты на содержание запасов.

Литература

1. Калинин Н.М., Хоботов Е.Н. Модели управления многопродуктовыми запасами при переменном спросе // Труды ИСА РАН. Динамика неоднородных систем. - М.: Издательство ЛКИ, 2008-Вып. 12 - С.185-200.

2. Линдерс М. Р., Фирон Х. Е. Управление снабжением и запасами. 4. Логистика // Перевод с англ. СПб.: ООО «Виктория плюс», 2002. 768 с.

3. Шрайбфедер Дж. Эффективное управление запасами. // Перевод с англ. 2-е изд. М.: Альпина Бизнес Букс, 2006. 304 с.

УДК: 519.72

ОБ ИНТУИЦИОНИСТСКИХ НЕЧЁТКИХ МНОЖЕСТВАХ

Г. Д. Коваль

Воронежский государственный университет

Введение

Необходимость обрабатывать информацию в условиях неопределенности привела к возникновению понятия нечёткого множества [1], принадлежность элементов к которому оценивается в промежутке $[0,1]$. Пусть U – универсальное множество, α – некоторое свойство, x – элемент U. Нечёткое подмножество (множество) A универсального множества U есть множество упорядоченных пар $A=\{(x/\mu_A(x))\}_{x\in U}$, где $\mu_A(x)\in[0,1]$ – степень принадлежности элемента $x\in U$ нечёткому множеству A.

Понятие нечёткого множества имеет как самостоятельное значение, так и используется, например, для определения нечётких чисел, термов лингвистических переменных. Нечёткое отношение является нечётким подмножеством специального универсального множества.

Значительный раздел теории нечётких множеств посвящен определению основных операций. Изначально для реализации объединения нечётких множеств использовалась операция $\max$, а пересечения – $\min$. Целенаправленный подход к определению нечётких операций стал возможен благодаря появлению треугольных норм и конорм [2]. Треугольная норма моделирует операции типа умножения (пересечение нечётких множеств, конъюнкция), а конорма − операции типа сложения (объединение нечётких множеств, дизъюнкция).

Треугольной нормой (*T-нормой*) называется операция $T:[0,1]\times[0,1]\to[0,1]$, удовлетворяющая следующим условиям [3,4]:

1) $T(x,y)=T(y,x)$ – коммутативность,
2) $T(T(x,y),z)=T(x,T(y,z))$ – ассоциативность,
3) $T(0,0)=0, T(x,1)=T(1,x)=x$ – ограниченность,
4) $(x\le t)\wedge(y\le z)\Rightarrow T(x,y)\le T(t,z)$ – монотонность.

Треугольной конормой (*S-конормой*) называется операция $S:[0,1]\times[0,1]\to[0,1]$, удовлетворяющая следующим условиям [3,4]:

1) $S(x,y)=S(y,x)$ – коммутативность,
2) $S(S(x,y),z)=S(x,S(y,z))$ – ассоциативность,
3) $S(1,1)=1, S(x,0)=S(0,x)=x$ – ограниченность,
4) $(x\le t)\wedge(y\le z)\Rightarrow S(x,y)\le S(t,z)$ – монотонность.

Легко заметить, что Т-нормы и S-конормы связаны соотношениями

$$T(x,y)=1-S(1-x,1-y) \text{ и } S(x,y)=1-T(1-x,1-y),$$

которые представляет собой соответствующие законы де Моргана с операцией стандартного отрицания $n(x)=1-x$. Совокупность операций (T,S,n) называется тройкой де Моргана.

Примеры треугольных норм и конорм можно найти в [2].

В настоящее время теория нечётких множеств представляет собой развитый математический аппарат, который широко и успешно используется для решения различных прикладных задач в условиях неопределенности.

В последнее время появились модификации и обобщения понятия нечёткого множества. Одно из них – понятие интуиционистского нечёткого множества. Его особенностью является то, что для каждого элемента $x \in U$ помимо степени принадлежности задается степень непринадлежности. Цели статьи заключается в представлении операций над интуиционистскими нечёткими множествами и их свойств. Интуиционистские нечёткие множества также набирают активность в приложениях, поэтому развитие данной теории является своевременным и актуальным.

1. Понятие интуиционистского нечёткого множества

Интуиционистским нечётким множеством (ИНМ) A называют множество вида

$$A = \left\{ \left(x / \mu_A(x), \nu_A(x) \right) \right\}_{x \in U},$$

где $x \in U$, $\mu_A(x) \in [0,1]$ – степень принадлежности элемента x интуиционистскому нечёткому множеству, $\nu_A(x) \in [0,1]$ – степень непринадлежности элемента x интуиционистскому нечёткому множеству, при этом для всех x из U имеет место неравенство $\mu_A(x) + \nu_A(x) \le 1$.

Заметим, что обычное нечёткое множество можно рассматривать как частный случай интуиционистского нечёткого множества при $\nu_A(x) = 1 - \mu_A(x)$ [3].

Для каждого интуиционистского нечёткого множества может определяться *интуиционистский индекс нечёткости*: $\pi_A(x) = 1 - \mu_A(x) - \nu_A(x)$, $x \in U$. Это выражение позволяет получить информацию о том, принадлежит ли x множеству U или нет. Очевидно, что $0 \le \pi_A(x) \le 1$ для $\forall x \in U$ [4].

2. Основные отношения и операции над интуиционистскими нечёткими множествами.

Для ИНМ A и B имеют место следующие отношения [3,4,5]:

$$A \subseteq B \Leftrightarrow \forall x \in U (\mu_A(x) \le \mu_B(x) \wedge \nu_A(x) \ge \nu_B(x)),$$
$$A = B \Leftrightarrow \forall x \in U (\mu_A(x) = \mu_B(x) \wedge \nu_A(x) = \nu_B(x)).$$

Важнейшая проблема – определение операций над интуиционистскими нечёткими множествами. В [3,4,5] основные операции определены следующим образом:

$$\overline{A} \Leftrightarrow \{x / \nu_A(x), \mu_A(x)\},$$
$$A \cap B = T(A(x), B(x)) = \{(x / \min(\mu_A(x), \mu_B(x)), \max(\nu_A(x), \nu_B(x)))\},$$
$$A \cup B = S(A(x), B(x)) = \{(x / \max(\mu_A(x), \mu_B(x)), \min(\nu_A(x), \nu_B(x)))\},$$
$$A \oplus B = \{x / \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x), \nu_A(x)\nu_B(x)\},$$
$$A \otimes B = \{x / \mu_A(x)\mu_B(x), \nu_A(x) + \nu_B(x) - \nu_A(x)\nu_B(x)\},$$
$$A - B = \{(x / \min(\mu_A(x), \nu_B(x)), \max(\nu_A(x), \mu_B(x)))\},$$
$$A \Delta B = \left\{ \begin{pmatrix} x / \max\left[\min\left(\mu_A(x), \nu_B(x)\right), \min\left(\mu_B(x), \nu_A(x)\right)\right], \\ \min\left[\max\left(\mu_A(x), \nu_B(x)\right), \max\left(\mu_B(x), \nu_A(x)\right)\right] \end{pmatrix} \right\},$$
$$A \times B = \{x / \mu_A(x)\mu_B(x), \nu_A(x)\nu_B(x)\}.$$

Заметим, что в приведенных определениях используются только классические операции $\max$ и $\min$.

Пример. Пусть заданы два ИНМ

$A_1 = \{(a/0.7,0.2),(b/0.4,0.3)\}$ и $A_2 = \{(a/0.7,0.1),(b/0.5,0.2)\}$, $a,b \in U$.

Выясним, в каких отношениях находятся данные множества и вычислим результаты введенных операций.

Так как для элемента a имеем $(0.7 \le 0.7 \wedge 0.2 \ge 0.1)$ и для элемента b $0.4 \le 0.5 \wedge 0.2 \ge 0.1$, то $A_1 \subset A_2$.

Найдем

$\overline{A_1} \Leftrightarrow \{(a/0.2,0.7),(b/0.3,0.4)\}$,

$\overline{A_2} \Leftrightarrow \{(a/0.1,0.7),(b/0.2,0.5)\}$,

$A_1 \cap A_2 = \{(a/\min(0.7,0.7),\max(0.2,0.1)),(b/\min(0.4,0.5),\max(0.3,0.2))\} =$

$= \{(a/0.7,0.2),(b/0.4,0.3)\}$,

$A_1 \cup A_2 = \{(a/\max(0.7,0.7),\min(0.2,0.1)),(b/\max(0.4,0.5),\min(0.3,0.2))\} =$

$= \{(a/0.7,0.1),(b/0.5,0.2)\}$,

$A_1 \oplus A_2 = \{(a/0.7+0.7-0.7*0.7,0.2*0.1),(b/0.4+0.5-0.4*0.3,0.3*0.2)\} =$

$= \{(a/0.91,0.02),(b/0.78,0.06)\}$,

$A_1 \otimes A_2 = \{(a/0.7*0.7,0.2+0.1-0.1*0.2),(b/0.4*0.5,0.3+0.2-0.3*0.2)\} =$

$= \{(a/0.49,0.28),(b/0.2,0.44)\}$,

$A_1 - A_2 = \{(a/\min(0.7,0.1),\max(0.2,0.7)),(b/\min(0.4,0.2),\max(0.3,0.5))\} =$

$= \{(a/0.1,0.7),(b/0.2,0.5)\}$,

$$A_1 \Delta A_2 = \begin{Bmatrix} (a/\max[\min(0.7,0.1),\min(0.7,0.2)],\min[\max(0.7,0.1),\max(0.7,0.2)]), \\ (b/\max[\min(0.4,0.2),\min(0.5,0.3)],\min[\max(0.4,0.2),\max(0.5,0.3)]) \end{Bmatrix} =$$

$= (a/0.2,0.7),(b/0.3,0.4)$,

$A_1 \times A_2 = \{(a/0.7*0.7,0.2*0.1),(b/0.4*0.5,0.3*0.2)\} = \{(a/0.49,0.02),(b/0.2,0.06)\}$.

Операции над интуиционистскими нечёткими множествами обладают рядом замечательных свойств. Рассмотрим некоторые из них.

Теорема 1 [5]**.** Пусть A и B являются ИНМ в непустом множестве X, тогда

1) $A - B = A \cap \overline{B}$;
2) $A - B = B - A$, если $A = B$;
3) $A - B = \overline{B} - \overline{A}$.

Доказательство.

1) Известно, что $A = \{(x/\mu_A(x),\nu_A(x))\}$, $B = \{(x/\mu_B(x),\nu_B(x))\}$, $x \in X$. $A - B = \{(x/\min(\mu_A(x),\nu_B(x)),\max(\nu_A(x),\mu_B(x)))\}$, но $\overline{B} = \{x/\nu_B(x),\mu_B(x)\}$, следовательно $A \cap \overline{B} = T(A(x),B(x)) = \{(x/\min(\mu_A(x),\nu_B(x)),\max(\nu_A(x),\mu_B(x)))\}$. Получим, что $A - B = A \cap \overline{B}$.

2) $A - B = \{(x/\min(\mu_A(x),\nu_B(x)),\max(\nu_A(x),\mu_B(x)))\}$. Если $A = B \Rightarrow \mu_A(x) = \mu_A(x)$ и $\nu_A(x) = \nu_B(x)$. Из этого следует, что $B - A = A - B$.

3) $A - B = \{(x/\min(\mu_A(x),\nu_B(x)),\max(\nu_A(x),\mu_B(x)))\}$. Учитывая, что $\overline{A} = \{x/\nu_A(x),\mu_A(x)\}$ и $\overline{B} = \{x/\nu_B(x),\mu_B(x)\}$, получим $\overline{B} - \overline{A} = \{(x/\min(\nu_B(x),\mu_A(x)),\max(\mu_B(x),\nu_A(x)))\}$.

Теорема 2 [5]**.** Пусть A и B являются ИНМ в непустом множестве X, тогда:

1) $A - A = \varnothing$;
2) $A - \varnothing = A$;
3) $A - B \subseteq A$;

4) $A-B=\varnothing$, если $A=B$;

5) $A-B=A$, если $B=\varnothing$ или $A\cap B=\varnothing$;

Приведённые выше утверждения легко доказываются.

Теорема 3 [5]**.** Пусть A, B и C являются ИНМ в непустом множестве X и $A\subseteq B\subseteq C$, тогда

1) $B-A\subseteq C-A$;

2) $B\Delta A\subseteq C\Delta A$.

Доказательство.

1) Учитывая, что $A\subseteq B\subseteq C$, т.е. $A\subseteq B$ и $B\subseteq C\Rightarrow A\subseteq C$, т.е. $\mu_A(x)\le\mu_B(x)\le\mu_C(x)$ и $\nu_A(x)\ge\nu_B(x)\ge\nu_C(x)$ $\forall x\in X$. Поскольку A является наименьшим по сравнению с B и C, то вычитание A из обеих частей $B\subseteq C$ ничего не изменит, т.е. $B-A\subseteq C-A$.

2) Поскольку "Δ" является расширением "$-$", то результат очевиден.

Следствие. Из основных операций можно вывести следующие соотношения:

1. $A\times B=B\times A$;
2. $(A\times B)\times C=A\times(B\times C)$;
3. $A\times(B\cup C)=(A\times B)\cup(A\times C)$;
4. $A\times(B\cap C)=(A\times B)\cap(A\times C)$;
5. $A\times(B\oplus C)=(A\times B)\oplus(A\times C)$;
6. $A\times(B\otimes C)=(A\times B)\otimes(A\times C)$.
7.

Заключение

Таким образом, обычное нечёткое множество можно рассматривать как частный случай интуиционистского нечёткого множества. Стоит отметить, что применение ИНМ вместо нечётких множеств предполагает введение дополнительных степеней свободы. Такое обобщение нечётких множеств дает дополнительную возможность представления недостаточных знаний о том, что лежит в описании многих действительных проблем. Также были определены основные отношения и операции над интуиционистскими нечёткими множествами.

Будущий объем исследовательской работы заключается в использовании треугольных норм и конорм для формализации операций над интуиционистскими нечёткими множествами и выявлении алгебраических свойств этих новых операций. Полученные результаты планируется применить для разработки методов принятия решений, в которых осуществляется обработка экспертной информации.

Список литературы

1. Zadeh L. A. Fuzzy sets // Information and Control. – Vol. 8 – 1965. – P. 338–353.

2. Klement E. P. Triangular norms. Position paper I: basic analytical and algebraic properties / E. P. Klement, R. Mesiar, E. Pap // Fuzzy Sets and Systems – Vol. 143. – 2004. – P. 5–26.

3. Леденева, Т. М.Обработка нечеткой информации: учебное пособие / Т. М. Леденева; Воронежский гос. ун-т. – Воронеж: Воронежский гос. ун-т, 2006. – 233 с.

4. Deschrijver G. On the Representation of Intuitionistic Fuzzy t-Norms and t-Conorms / G. Deschrijver, C. Cornelis, E. E. Kerre // IEEE. – 2004. – Vol. 12, № 1. – P. 42–61.

5. Ejegwa P. A. Intuitionistic Fuzzy Set and Its Application in Career Determination via Normalized Euclidean Distance Method / P. A. Ejegwa, A. J. Akubo, O. M. Joshua // European Scientific Journal. – 2014. – Vol. 10, № 15. – P. 539–536.

УДК 004

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ПОИСКА ПОТЕРЯННЫХ ДОМАШНИХ ЖИВОТНЫХ

Колтаков И.П.

Воронежский государственный университет

Введение

Домашние животные всегда играли важную роль в жизни их хозяев. Но даже у самых ответственных людей может потеряться любимый питомец. В таком случае актуальными инструментами поиска становятся информационные технологии. Существует множество различных решений. Чаще всего приходится сталкиваться с перегруженными приложениями, где крайне сложно осуществить поиск нужной информации ввиду отсутствия её декомпозиции, и пользователю приходиться самостоятельно проводить сортировку. Большинство приложений не реализует все необходимые способы и методы, которые бы значительно упростили жизнь пользователей.

Таким образом, возникает необходимость в создании приложения, которое позволило бы разместить публикацию о потерянном животном, чтобы всю идентифицирующую информацию о нём предоставить другим участником сервиса, что может так или иначе помочь в поисках.

1. Популярные решения

Рассмотрим наиболее часто используемые технологии, пользующиеся популярностью у людей, попавших в вышеупомянутую ситуацию. Результаты сравнения представлены в табл.

Сравнение существующих решений

Критерий	«ВКонтакте» [1]	«Авито» [2]	«POISKZOO.RU» [3]
Структурированность информации	-	+	-
Коммуникация с автором объявления	+	-	-
Удобство добавления объявления	-	-	-
Система фильтрации	-	+	-
Внешний вид	-	+	-

На основании проведенного анализа можно сделать вывод, что актуально создание собственного приложения, позволяющего реализовать многокритериальный поиск животного, получение, добавление, обновление, удаление, размещение объявлений, возможность загрузки фотографий, добавление комментариев для каждого объявления. Это поможет сделать продукт максимально удобным.

2. Технические средства

Для реализации приложения были использованы следующие средства: язык разработки C#, платформа ASP.NET, Blazor web assembly, библиотека MudBlazor, Entity Framework для взаимодействия с базой данных, среда разработки Visual Studio 2022, СУБД PostgreSQL, сервер приложений Docker.

В качестве среды разработки была выбрана среда разработки Visual Studio, так как она обладает множеством полезных функций, упрощающих написание кода.

Для работы с базой данных был выбран PostgreSQL, потому что он позволяет создать объектную модель базы данных и работать с обычными entity-классами вместо написания запросов.

3. Пример работы приложения

На рис. 1 представлен фрагмент страница объявления. При входе на страницу пользователь видит боковое меню и объявления. На каждом объявлении есть 5 кнопок, нажав на которые пользователь может открыть объявление, добавить его в избранное, просмотреть список комментариев, добавить свой комментарий и оформить подписку на выбранное им объявление. При оформлении подписки пользователь разрешает приложению отправку электронных писем, которые оповещают его о новых комментариях или изменениях в объявлении. Такая функциональная отличает разработанное приложение от аналогичных.

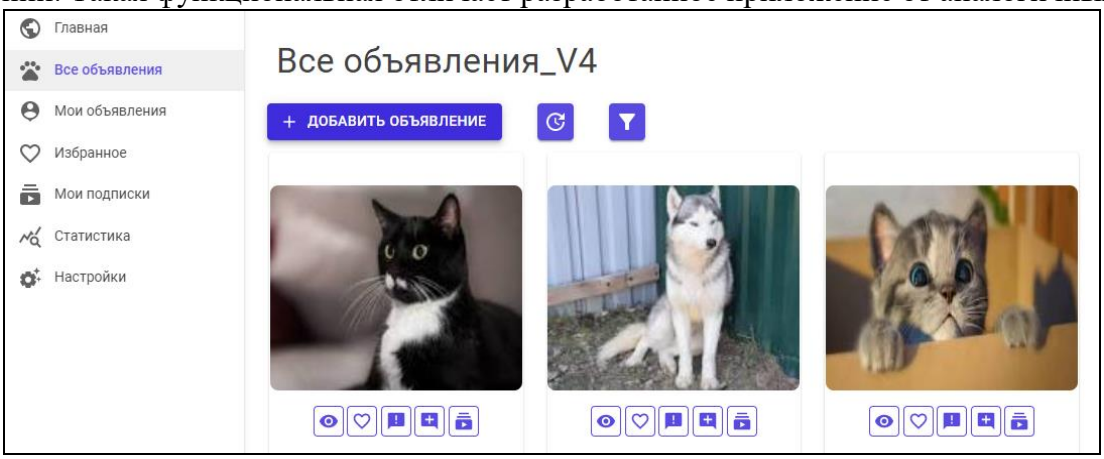


Рис. 1. Страница объявлений

На рис. 2 и 4 представлены фрагменты диалогового окна добавления объявления и добавления фильтра для поиска, где пользователь может указать всю необходимую информацию о животном. Приложение ориентировано именно на поиск потерянных питомцев, поэтому предоставляет пользователям только необходимые фильтры для добавления информации и животном, а идентичность структуры объявлений позволяет обеспечить удобный просмотр и быстрый поиск.

Рис. 2. Второй фрагмент диалогового окна добавление объявления

Важно отметить, что пользователю не нужно беспокоиться о формате изображения, так как приложение принимает все типы изображений. Одним из преимуществ является упрощение добавления объявления. Для заполнения обоих диалоговых окон задействуется минимальное количество текстовых полей, в большинстве случаев пользователю нужно выбрать нужный ему вариант из представляемого ему списка (рис. 3).

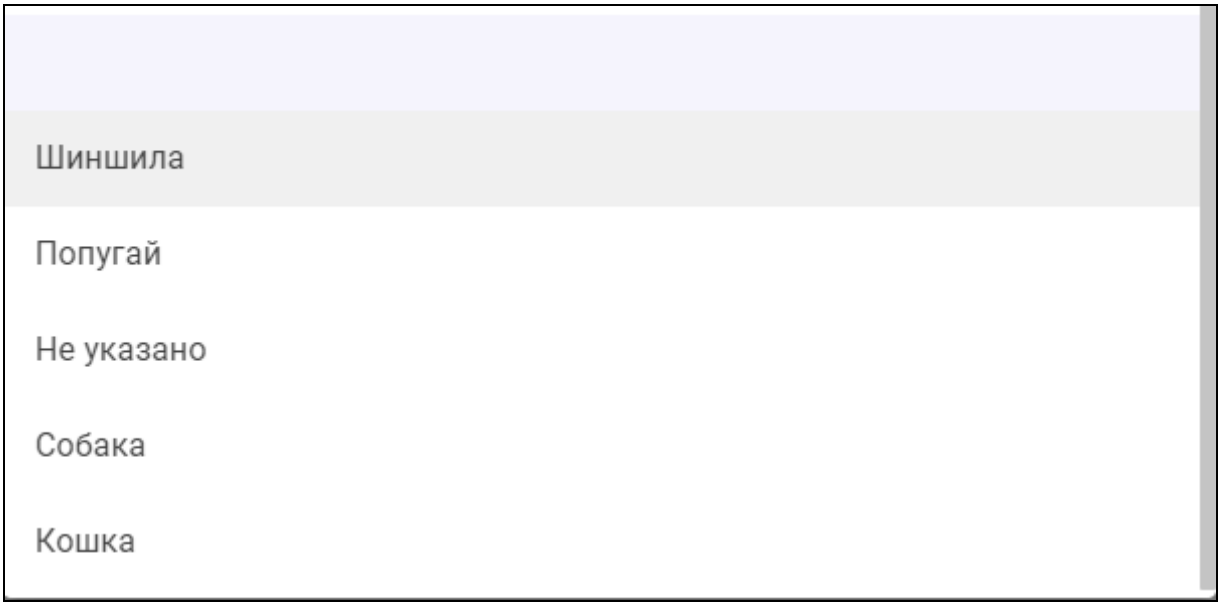


Рис. 3. Представляемый пользователю список

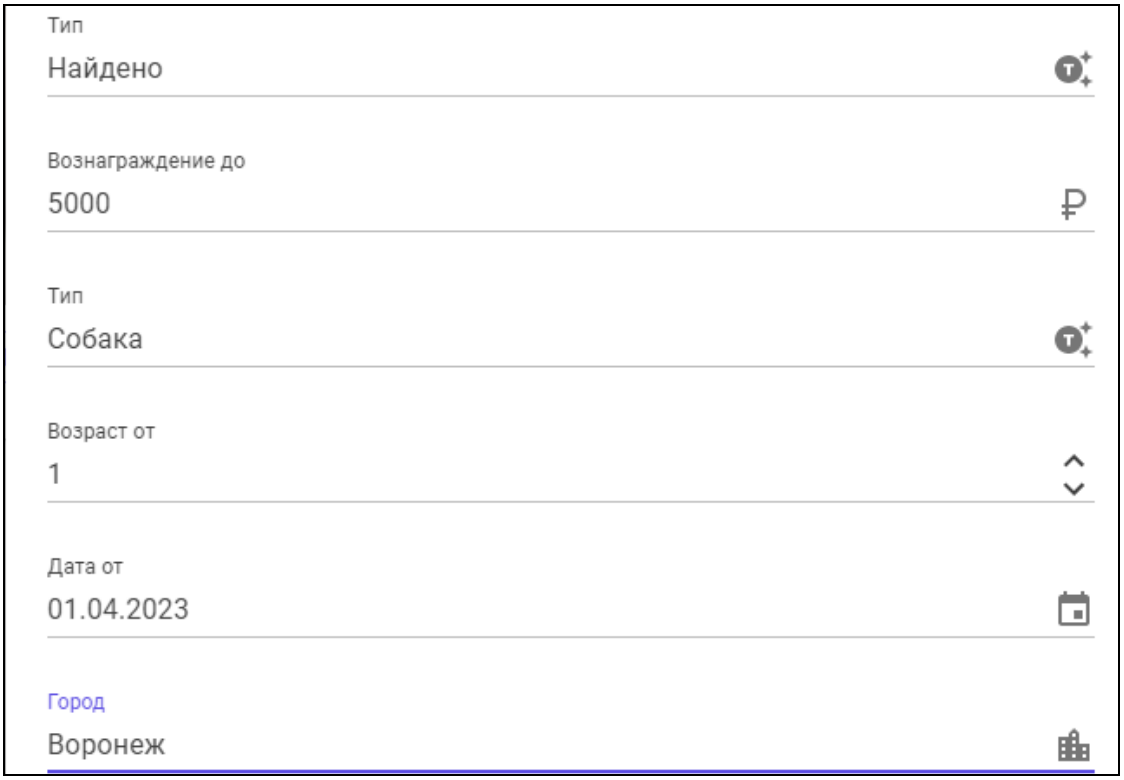


Рис. 4. Фрагмент диалогового окна добавления фильтра для выдачи нужных объявлений

Заключение

Созданное приложение должно помочь владельцам потерявшихся домашних животных, обеспечивает максимально возможную простоту использования, скорость и нацеленность поиска, позволяет привлечь к поиску большее количество людей, а также помогает тем людям, кто нашёл питомца, но не представляет как сообщить об этом хозяину.

Литература

1. ВКонтакте. – Режим доступа: https://vk.com/lost_vrn – (Дата обращения: 10.03.2023).
2. Авито: https://www.avito.ru/ – (Дата обращения: 04.03.2023).
3. POISKZOO.RU: https://poiskzoo.ru/ – (Дата обращения: 02.04.2023).
4. Размещение и развертывание ASP.NET Core Blazor WebAssembly – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/blazor/ – (Дата обращения: 10.01.2023).
5. Исследование MudBlazor WebAssembly – Режим доступа: https://mudblazor.com/docs/overview – (Дата обращения: 26.01.2023).
6. Зачем использовать PostgreSQL – Режим доступа: https://www.postgresql.org/about/ – (Дата обращения: 07.02.2023).
7. Entity Framework – Режим доступа https://metanit.com/sharp/efcore/7.3.php – (Дата обращения: 12.01.2023).

УДК 004.42

РАЗРАБОТКА ANDROID-ПРИЛОЖЕНИЯ «ЗООМАГАЗИН»

А.В.Коржова

Воронежский Государственный Университет

Введение

В статье проводится анализ разработки мобильного приложения – магазин. Рассматриваются современные методы решения, требования к реализации. Осуществляется выбор оптимального метода для разработки приложения.

1.Реализация задумки

Сейчас интернет является одним из главных инструментов развития бизнеса. Основное использование интернета для продвижения бизнеса являются онлайн продажи.

Торговлю в интернете осуществляют с помощью интернет-магазинов. Магазины могут быть разной классификации по модели бизнеса, по товарному ассортименту. При такой деятельности продавец и покупатель при осуществлении торговых отношений не обязательно должны находится в одном конкретном месте. Оплата может проходить через виртуальную платежную систему, а доставка товара с помощью курьерской службы или почты.

Интернет-магазин имеет ряд преимуществ: экономия времени, большой выбор ассортимента, быстрый поиск нужного товара, экономия средств. Что касается недостатков: нельзя потрогать товар (хотя сейчас возможность медиа-технологий позволяет осмотреть товар со всех сторон), проблемы гарантии, зачастую долгая доставка, риск столкновения с поддельным товаром.

Существует два вида интернет-магазинов: веб-сайт и приложение на мобильное устройство. Обычно владельцы магазина внедряют и веб-приложение и мобильное приложения для удобства клиентов. Далее будет рассматриваться разработка мобильного-приложения, но проектирование достаточно схоже с разработкой веб-приложений, за исключением использования разных инструментов реализации.

Важно упомянуть компоненты интернет магазина:

1. Интернет витрина;
2. Система приема платежей;
3. Информационные системы.

Процесс разработки интернет-магазина в общем схож с процессом разработки любого другого мобильного приложения, однако имеет специфику, т.к. является средством ведения торговли.

Этапы проектирования магазина:

1. Выбор СУБД – сервера;
2. Организация информационного наполнения(контента);
3. Разработка систем навигации;
4. Разработка системы оплаты и доставки;

Остановимся на одном специфическом моменте в проектировании при создании зоомагазина – информационное наполнения.

Необходимо сформировать сведения для работы с мобильным приложением: структура приложения, правила пользования, порядок регистрации и оформления товара, порядок оплаты заказа, доставки и возврата.

Самое важное, это грамотное описание характеристики товара. Подбор верных свойств товара, безошибочных внос товара в базу (например, граммовка), распределение товаров в нужные категории, распределение товаров для каждого питомца. Для демонстрации товара используется фото с различных ракурсов. Так же можно добавить дополнительную информацию к товару: сведения о производителе, инструкция, ролик с демонстрацией товара в действии (например, игрушки для питомцев), рецепты рациона от ветеринара (корм, лакомства).

Для интернет-магазина наиболее важным является структурирование информации, а добавление дополнительных сведений к карточкам товара предотвратит не верную покупку у клиентов и возврат товара.

Таким образом, для удовлетворения всех требований к Android-приложению планируется реализация приложения “My Pet`s”, которое позволит совершать удобные покупки для питомцев и перевозку животных по городу и между городов.

Для создания ПО необходимо определить этапы работы, отражающие задачи, решить которые будет необходимо при реализации:

1. Анализ существующих аналогов продукта;
2. Разработка требований к приложению;
3. Дизайн;
4. Разработка самого ПО;
5. Тестирование ПО;
6. Подготовка к эксплуатации – устранение ошибок.

К любому программному продукту должны быть предъявлены требования, которые он должен выполнять. Интернет магазин по типу относится к коммерческой разработке, основные задачи которого:

- регистрация/выполнение входа в аккаунт;
- выбор категорий товаров, а также определенного животного;
- удобный поиск товаров;
- предоставление информации о товарах, услугах и производителе;
- доставка;
- перевоз питомцев;
- предоставление онлайн помощи клиенту (консультации);
- предоставление необходимой дополнительной информации относительно товаров, услуг.
- работа с электронной корзиной;
- оформление заказа с выбором доставки, оплаты;
- обеспечение безопасности личной информации клиента;
- удобство использования – дружелюбный интерфейс пользователя;
- расширяемость – возможность добавления новых функций.

Особенности мобильного приложения по сравнению с существующими аналогами:

1. Бесплатный доступ к рационам питания от ветеринаров;
2. Помощь по поиску пет-сервисов;
3. Сервис «Такси» для животных, в случае необходимости отправить питомца в любую точку города или в другой город.

Для разработки приложения будут использоваться следующие инструменты: объектно-ориентированный высокоуровневый язык программирования Java, IDE Android Studio, система

управления реляционными базами данных SQLite, программа для создания, проектирования и документирования баз данных ERWin, а также шаблон проектирования MVVM.

По MVVM приложение разделяется на три главных компонента [1]:

- View – содержит поля, соответствующие интерфейсу пользователя;
- ViewModel – те же поля, но в предметной области;
- Сама модель (Model).

Это позволяет связывать элементы View со свойствами и событиями ViewModel. При этом ViewModel — абстракция представления. Свойства View совпадают со свойствами ViewModel/Model. При этом ViewModel не имеет ссылки на интерфейс представления. Изменение состояния ViewModel автоматически изменяет View, и наоборот. Для этого используется механизм связывания данных. Также характерная черта MVVM — двусторонняя коммуникация с View[2].

В работе используется базовый подход, который заключается в создании пустого проекта и постепенного добавления всех необходимых конфигураций, файлов и компонентов.

Для правильного программирования приложения были придуманы классы(рис.1):

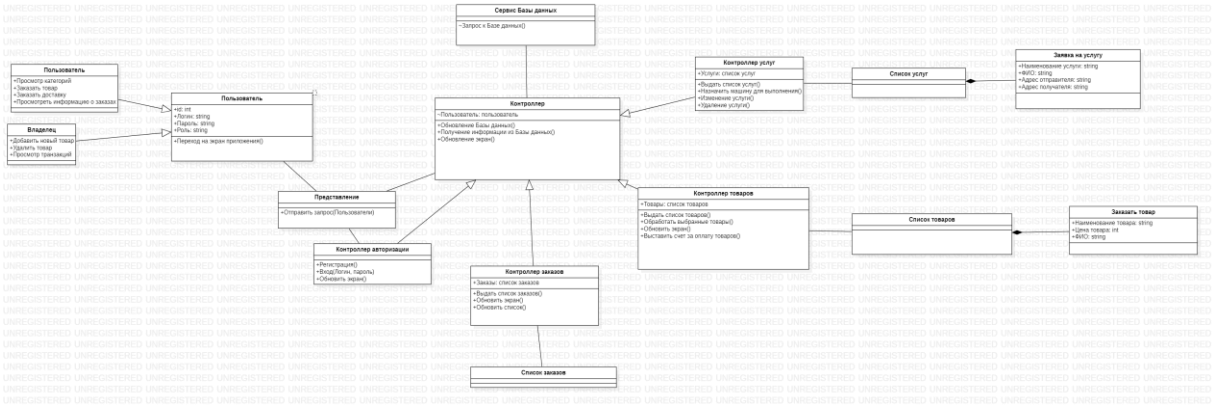


Рис.1.Диаграмма классов

Для понимания работы приложения со стороны клиента и со стороны администратора (т.е. владельца магазина) нижи приведена use-case диаграмма(рис.2).

Если подытожить, то со стороны клиента можно выполнять следующие операции:

1. Вход/регистрация;
2. Просмотр каталога;
3. Выбор товара;
4. Оформление заказа;
5. Вызов доставки.

Со стороны администратора (т.е. владельца магазина) можно выполнить следующее:

1. Вход/регистрация;
2. Добавление нового товара/новой категории;
3. Просмотр заказов.

Всего приложение имеет десять экранов со стороны пользователя:

1. для входа – экран для выполнения входа в учетную запись;
2. для регистрации – экран для создания новой учетной записи;
3. с категориями товаров по виду домашнего питомца – на экране перечислены животные: собака, кошка, грызуны, другие животные, пет-сервисы, при нажатии на ту или иную категорию происходит переход на следующий экран;
4. с товарами для собак – экран, где перечислены все различные товары для собак;
5. с товарами для кошек – экран, где перечислены все различные товары для кошек;

6. с товарами для грызунов – экран, где перечислены все различные товары для грызунов;
7. с товарами для других животных – экран, где перечислены все различные товары для рыбок, птиц и т.п.;
8. с пет-сервисами – экран, где перечислены адреса и контакты проверенных вет-клиник;
9. личный кабинет – экран с информацией о пользователе и доступом к заказам;
10. корзина;
11. доставка/такси – экран, где можно заказать такси для питомца или доставку из магазина.

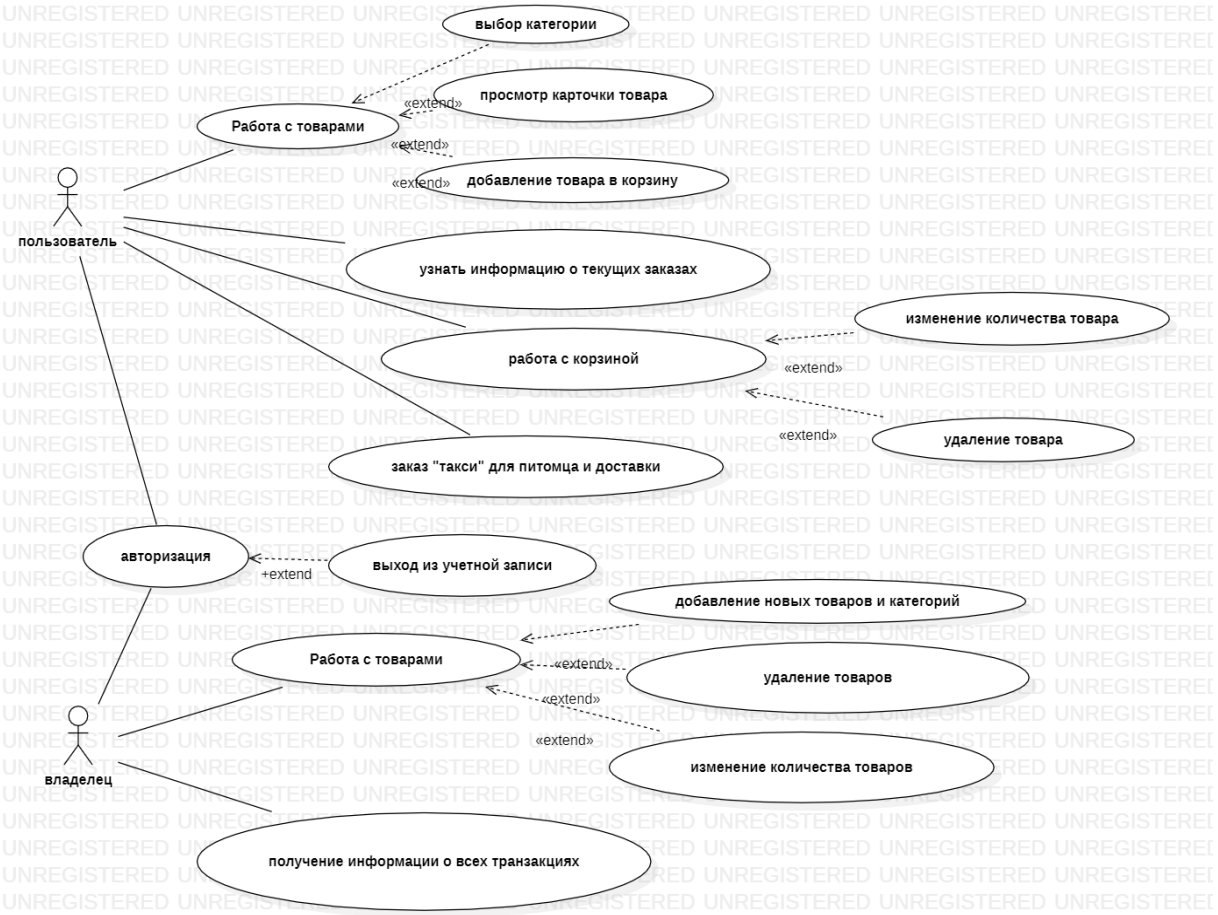


Рис.2. Use-case диаграмма

И три экрана со стороны администратора:
1. для входа – экран для вхождения в учетную запись;
2. с перечислением всех товаров – экран, где видна таблица с товарами, необходимо для отслеживания количество товаров и добавления новых позиций;
3. с транзакциями – экран, где видна таблица со всеми заказами.

Для того, чтобы хранить товары и их описание всех перечисленных выше категорий необходима база данных. Для работы с Android было выбрано СУБД SQLite[3]. База данных будет иметь следующий вид(Рис.3).

Небольшое описание каждой таблицы базы:
1. Clients – табличка для хранения информации о клиентах.

2. Personal – табличка для хранения информации об администраторах.
3. Types– табличка для хранения категорий товаров.
4. Provider – табличка для хранения информации о производителях товаров.
5. Goods – табличка для хранения всех товаров. Содержит поля: id, наименование, цена, описание, производитель.

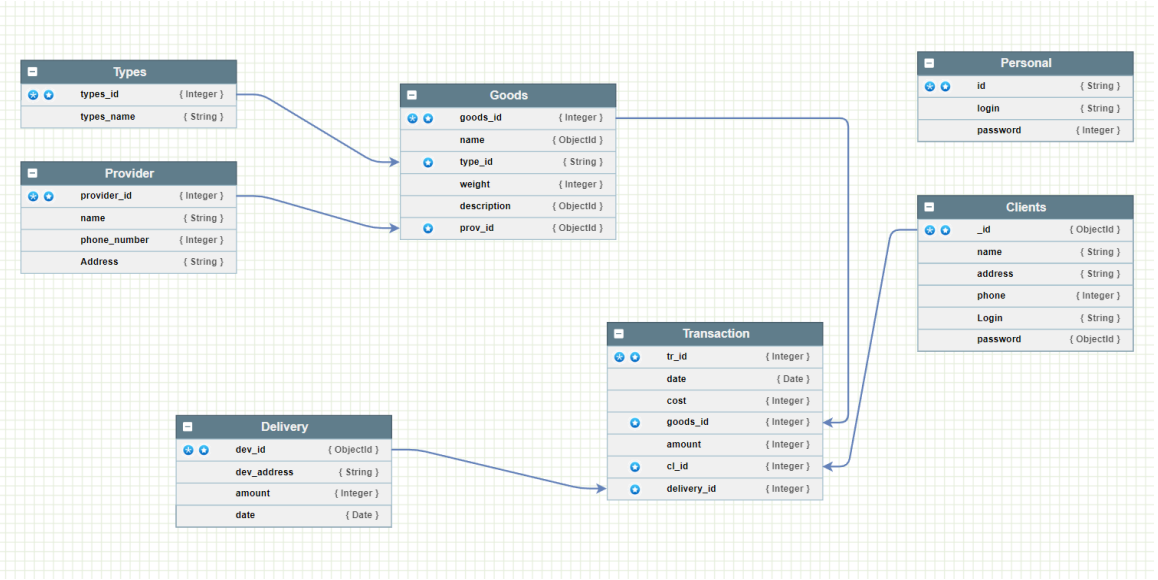


Рис.3. Схема Базы данных

6. Transaction – табличка для хранения информации о заказах
7. Delivery – табличка для хранения информации о доставке

Данную разработку можно будет считать основой мобильного приложения-зоомагазина, которую можно будет удобно совершенствовать и развивать по мере необходимости.

Приложение так же имеет огромный потенциал в расширении, что повышает его ценность и полезность:

1. перевод приложения на другие языки
2. расширение сети, добавление новых товаров и категорий;
3. добавление сервисов для подсчета потребляемой питомцем еды;
4. работа приложения на платформе IOS.

Заключение

Исходя из поставленной цели разработать мобильное приложение можно сказать, что было выполнено: рассмотрение существующих интернет магазинов, их достоинства и недостатки; обоснование создание мобильного интернет-магазина; выбор технология для разработки; разработка приложения и базы данных для магазина.

Литература:

1. Android Developers // URL: https://developer.android.com
2. Использование шаблона MVVM // URL: https://habr.com/ru/companies/dataart/articles/272737/
3.Дейтл П., Дейтл Х., Уолд А., Android для разработчиков. //3-е издание – СПб.: Питер, 2016. – 512 с.

УДК 004.031.42

МОДЕЛЬ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ НА ОСНОВЕ ЛИНГВИСТИЧЕСКОГО ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

А.И. Косарыч

Воронежский государственный университет

Введение

Наиболее распространенным типом интеллектуальных информационных систем являются рекомендательные системы, получившие особенно широкое распространение в области электронной торговли. Существует четыре основных типа рекомендательных систем, базирующихся на следующих технологиях: фильтрация, основанная на контент; коллаборативная фильтрация; фильтрация, основанная на знаниях; гибридные рекомендательные системы. Рекомендательные системы помогают пользователям находить подходящие товары с помощью рекомендаций, основанных на информации из различных источников, при этом в большинстве случаев информация является числовой. Однако для повышения степени выразительности, а, следовательно, и точности предлагаемых рекомендаций целесообразно использовать лингвистическую модель, которая актуальна для обработки экспертной информации. Именно такая информация, полученная от пользователей, обрабатывается в рекомендательных системах. К основным процессам в рекомендательных системах относятся: *процесс профилирования*, направленный на создание профиля пользователя – информационной структуры, которая выражает предпочтения пользователя на основе представленных им примеров; *процесс разработки рекомендаций* – система измеряет сходство между профилем пользователя и элементами из базы данных о товарах, а затем ранжирует эти объекты в соответствии с показателем схожести и рекомендует наиболее схожие. Цель статьи заключается в разработке алгоритмов, реализующих указанные процессы.

1.Алгоритм формирования профиля

На данном этапе система собирает предпочтения пользователя, чтобы знать, какой для него требуется товар. Пусть рекомендательная система имеет базу данных, в которой товары-объекты из множества $A=\{a_1,...,a_m\}$ описаны с помощью набора атрибутов $C=\{c_1,...,c_n\}$ так, что каждому объекту a_j соответствует векторная оценка $f_j = f\left(a_j\right)=\left(v_1^j,...,v_n^j\right)$, $j=\overline{1,m}$, при этом каждая частная оценка формируется в некоторой лингвистической шкале S_k, включающей $k+1$ термов в форме нечетких чисел определенного типа. Лингвистические шкалы, используемые для формирования базы данных, будем называть базовыми. Количество термов в шкале определяет ее гранулярность. Чем больше гранулярность, тем лингвистические оценки являются менее расплывчатыми и более точными в некотором смысле. Векторную оценку f_j будем называть вектором полезности товара a_j. Данные описания могут бы получены либо от экспертов, либо на основе опросов, проведенных в отношении этих товаров. Заметим, что каждый атрибут может быть оценен в «своей» лингвистической шкале в соответствии с существующей степенью знаний о нем. Таким образом, база знаний содержит лингвистическое описание товаров, из которых пользователь выбирает нужный ему.

1.1 Формирование начального профиля

Чтобы построить профиль пользователя, необходимо сформировать его систему предпочтений на основе выбора примеров предпочтительных товаров-объектов. Эти примеры образуют множество, которое можно рассматривать как обучающее. Пусть $U = \{u_1, ..., u_k\}$ – множество предпочтительных примеров товаров, выбранных пользователем, и для некоторого i имеем $u_i = a_j$, т.е. в качестве примера выступает товар a_j. Этот элемент описан в базе данных с помощью вектора полезности $f_j = f(a_j) = (v_1^j, ..., v_n^j)$, в котором $v_l^j \in S_k$, k – количество термов в лингвистической шкале. Данный пример определяет начальный профиль пользователя, который мы обозначим как $P_r^0 = (p_1^r, ..., p_n^r)$, где $p_s^r = v_s^j$ для всех $s = \overline{1,n}$, $r \in \{1, ..., L\}$, L – количество предпочтительных примеров, указанных пользователем $P^0 = \{P_1^0, ..., P_L^0\}$. Таким образом, указывая на предпочтительные товары, пользователь формирует свой начальный профиль P^0, при этом описания примеров совпадают с теми, которые имеются в базе данных.

1.2 Модификация начального профиля

После определения начального профиля пользователя P^0 система предоставляет ему возможность изменять одно или несколько значений его профиля в целях совершенствования процесса вынесения рекомендаций. Вероятно, знания пользователя о заданном атрибуте предпочтительного примера отличаются от знаний специалистов о нем, т.е. лингвистическая оценка какого-либо атрибута предпочтительного примера требует уточнения от пользователя с использованием другой лингвистической шкалы. В этом случае система предоставляет пользователю возможность использовать другой набор лингвистических термов, более подходящий для выражения его знаний об атрибуте предпочтительного примера. Модификация начального профиля позволяет уточнить предпочтения пользователя и на их основе выдать наиболее релевантные рекомендации. Таким образом, в начальном профиле для некоторых предпочтительных примеров пользователь может изменить частные оценки, используя другие лингвистические шкалы, отличающиеся от базовых степенью гранулярности, т.е. количеством используемых лингвистических термов. При переходе к другой лингвистической шкале описание примеров из начального профиля меняется с учетом новой лингвистической шкалы, используемой пользователем. Тем самым, будет сформирован новый профиль пользователя $\tilde{P} = \{\tilde{P}_1, ..., \tilde{P}_L\}$, где в каждом примере $\tilde{P}_r$ частная оценка $\tilde{p}_j^r$ будет изменена на оценку в лингвистической шкале $\tilde{S}_d$ с количеством термов d. Если пользователь не изменил оценку, то $\tilde{p}_j^r = p_j^r$.

1.3 Унификация лингвистических шкал

В результате модификации начального профиля будет сформирован индивидуальный профиль пользователя, включающий совокупность примеров (товаров), которые соответствуют предпочтениям пользователя в наибольшей степени, при этом оценки атрибутов будут оценены в лингвистических шкалах с различным количеством термов. В различных шкалах один и тот же терм имеет различный семантический смысл, так что информация для выработки рекомендаций является разнородной. Понятно, что для обработки такой информации и

обеспечения сопоставимости ее необходимо унифицировать, т.е. привести к некоторой универсальной шкале. *Универсальная лингвистическая шкала* может быть выбрана из уже существующих или построена, но в любом случае ее размерность должна быть не меньше, чем размерность уже используемых для оценки шкал.

Для перехода из шкалы S_j в универсальную шкалу S_U будем использовать *функцию трансформации* $\tau_{S_j \to S_U} : S_j \to F(S_U)$, такую что

$$\tau_{S_j \to S_U}(S_i^j) = \left\{ \left(S_k^U / \alpha_k^i \right) \right\}_{k=\overline{0,N}},$$

$$\alpha_k^i = \max_y \min \left\{ \mu_{S_i^j}(y), \mu_{S_k^U}(y) \right\},$$

где $\mu_{S_i^j}(y), \mu_{S_k^U}(y)$ – функции принадлежности нечетких множеств, определяющих термы $S_i^j \in S^j$ и $S_k^U \in S_U$ соответственно, $F(S_U)$ – нечеткое подмножество универсальной лингвистической шкалы S_U.

Пусть эксперт E_j в процедуре оценивания использует шкалу $S_j = \left\{ S_0^j, S_1^j, S_2^j, S_3^j, S_4^j \right\}$, которая содержит 5 термов, причем каждому терму соответствует нечеткое треугольное число

$$S_0^j = (0,0,0.25), S_1^j = (0,0.25,0.5), S_2^j = (0.25,0.5,0.75), S_3^j = (0.5,0.75,1), S_4^j = (0.75,1,1).$$

Предположим, что универсальная шкала содержит 7 термов

$$S_U = \left\{ S_0^U, S_1^U, S_2^U, S_3^U, S_4^U, S_5^U, S_6^U, S_7^U \right\}$$

также с треугольными функциями принадлежности

$$S_0^U = (0,0,0.16), S_1^U = (0,0.16,0.34), S_2^U = (0.16,0.34,0.5), S_3^U = (0.34,0.5,0.66),$$

$$S_4^U = (0.5,0.66,0.84), S_5^U = (0.66,0.84,1), S_6^U = (0.84,1,1).$$

После применения преобразования $\tau_{S_j \to S_T}$ получим нечеткое подмножество шкалы S_U. Например, для термов S_0^j и S_1^j функции принадлежности будут иметь следующий вид:

$$\tau_{S_j \to S_U}\left(S_0^j\right) =$$

$$= \left\{ \left(S_0^U / 1\right), \left(S_1^U / 0.58\right), \left(S_2^U / 0.18\right), \left(S_3^U / 0\right), \left(S_4^U / 0\right), \left(S_5^U / 0\right), \left(S_6^U / 0\right) \right\},$$

$$\tau_{S_j \to S_U}(S_1^j) =$$

$$= \left\{ \left(S_0^U / 0.39\right), \left(S_1^U / 0.85\right), \left(S_2^U / 0.85\right), \left(S_3^U / 0.39\right), \left(S_4^U / 0\right), \left(S_5^U / 0\right), \left(S_6^U / 0\right) \right\}.$$

На рис.1 изображены обе шкалы S^j (сплошная линия) и S^U (пунктирная линия).

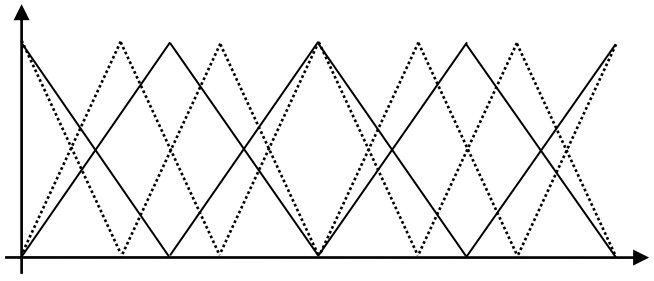


Рис. 1 Шкалы S^j и S^U

На рис. 2 представлен график функций принадлежности полученного нечеткого множества для терма S_0^j.

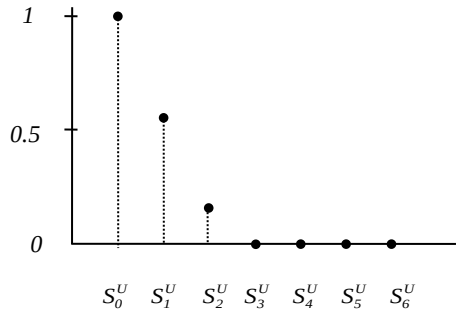


Рис.2 График функции для терма S_0^j

Будем считать, что универсальная лингвистическая шкала представляет собой множество термов $S^U = \left\{S_0^U, ..., S_N^U\right\}$, где $N+1$ – размерность лингвистической шкалы.

2. Алгоритм формирования рекомендаций

На этой стадии система вычисляет, насколько близки товары-объекты к профилю пользователя, с помощью измерения сходства или сходства. Для выполнения этого этапа система оценит сходство между всеми элементами базы данных $A = \left\{a_1, \ldots, a_n\right\}$ и профилем пользователя после следующих шагов:

1. *Вычисление сходства между каждым объектом и профилем пользователя:* система вычисляет степень сходства между профилем пользователя и элементами базы данных.

2. *Предоставление рекомендаций:* наконец, система предлагает пользователю наиболее подходящие товары-объекты, то есть наиболее близкие к профилю пользователя.

2.1 Вычисление сходства между профилем пользователя и товарами из базы данных

Пусть $a_j \in A$ – некоторый товар-объект из базы данных, ему соответствует векторная оценка $f_j = f\left(a_j\right) = \left(v_1^j, ..., v_n^j\right)$, где каждая частная оценка v_i^j есть нечеткое подмножество универсальной лингвистической шкалы вида $\left\{\mu_{v_i^j}\left(k\right) / S_k^U\right\}_{k=\overline{0,N}}$. Выберем товар $\tilde{P}_r$ из профиля пользователя. В универсальной лингвистической шкале каждая частная оценка $\tilde{p}_l^r$ также представляет собой нечеткое подмножество $\left\{\mu_{\tilde{p}_l^r}\left(k\right) / S_k^U\right\}_{k=\overline{0,N}}$. Здесь $\mu_{v_i^j}\left(k\right)$ и $\mu_{\tilde{p}_l^r}\left(k\right)$ – степени принадлежности соответствующих оценок терму S_k^U. Задача сводится к определению степени сходства или несходства данных нечетких множеств.

Заметим, что для вычисления данных характеристик можно использовать различные метрики, в том числе известные функции расстояния.

Пусть A и B – нечеткие множества, определенные на одном и том же универсальном множестве U и имеющие функции принадлежности μ_A и μ_B. В наиболее общей форме функция расстояния представляется формулой Минковского

$$\rho(A,B)=\left(\sum_{u\in U}\left|\mu_A(u)-\mu_B(u)\right|^{\alpha}\right)^{1/\alpha}.$$

Ее особенностью является наличие параметра, который можно настраивать на обучающей выборке. Частными случаями данной формулы являются расстояние Хемминга $(\alpha=1)$ и расстояние Евклида $(\alpha=2)$.

2.2 Разработка рекомендаций

На данном этапе система ранжирует товары согласно значениям коэффициентов сходства. Лучшими будут те, которые ближе к профилю пользователя, то есть те, которые имеют наибольшее значение коэффициента сходства. Система будет рекомендовать топ-N товаров, которые достигают заданного порога, т.е. если один из топ-N товаров слишком далек от профиля пользователя (его степень сходства меньше порога), то этот товар не будет включен в рекомендацию.

Заключение

В данной статье был представлен алгоритм рекомендательной системы на основе лингвистического представления информации, причем такая информация формируется пользователем в индивидуальной лингвистической шкале, которая затем преобразуется в универсальную шкалу. Преимущество такого представления заключается в том, что мы можем собрать информацию пользователя, которая обычно связана с восприятием или вкусами, без потери выразительности или точности. Кроме того, сформирована гибкая модель работы с информацией, в которой каждый атрибут может быть оценен с помощью наиболее подходящего набора лингвистических термов, а пользователи могут использовать наборы лингвистических термов в соответствии со своими знаниями или предпочтениями.

Литература

1. Кокачев, В. А. Рекомендательные системы в контексте технологий больших данных: дис. Санкт-Петербургский гос. университет, Санкт-Петербург, 2018. — Режим доступа: https://dspace.spbu.ru/bitstream/11701/12104/1/Kokachev_V.pdf

2. L. Martinez, Manuel J. Barranco, Luis G. Perez, Macarena Espinilla A Knowledge Based Recommender System with Multigranular Linguistic Information: International Journal of Computational Intelligence Systems, Vol.1, No. 3 (August, 2008), 225 – 236

3. Леденева, Т. М. Обработка нечеткой информации: учебное пособие /Т.М. Леденева. – Воронеж: Воронежский государственный университет, 2006. – 235 с.

УДК 004.056.5+004.8

ПРИМЕНЕНИЕ CHATGPT ДЛЯ ВЫЯВЛЕНИЯ НОВЫХ УГРОЗ БЕЗОПАСНОСТИ

Г. Ю. Котляров, В. В. Бутузов

Воронежский государственный университет

Введение

С развитием технологий обработки естественного языка (Natural Language Processing, NLP) возникают новые угрозы в области безопасности. К примеру, вредоносные атаки могут использовать модели NLP для создания фишинговых писем или для обхода систем аутентификации. Помимо этого, злоумышленники могут использовать модели NLP для генерации текстов, содержащих ненормативную лексику или дискриминационные высказывания, что может нанести вред репутации компаний и индивидуальных пользователей.

Одной из наиболее эффективных технологий NLP для выявления угроз безопасности является использование моделей генерации текстов на основе нейронных сетей, в частности, модели ChatGPT [1]. Эти модели могут использоваться для анализа текстов на предмет наличия угроз и для создания новых алгоритмов защиты, таких как системы обнаружения аномалий и системы распознавания текстовых атак.

Цель данной статьи заключается в описании применения модели ChatGPT для выявления новых угроз безопасности в текстовых данных. Для достижения этой цели были изучены и проанализированы существующие исследования в области применения моделей NLP в кибербезопасности, включая статьи [1-7]. Для достижения цели статьи были проанализированы эксперименты с использованием модели ChatGPT и датасетов, содержащих тексты с различными угрозами безопасности.

1. Обзор методов NLP в кибербезопасности

1.1. Краткое описание методов анализа текста в кибербезопасности

Анализ текста является важным инструментом в области кибербезопасности, так как большое количество информации связано с текстовыми данными, например, логами систем и мониторингом социальных сетей. Среди методов анализа текста, которые используются в кибербезопасности, можно выделить следующие:

1. Классификация текста: это процесс определения категории, в которую может быть отнесен текст (например, спам или не спам). Классификация текста может быть использована для обнаружения атак, таких как фишинг и мошенничество [3].
2. Извлечение информации: это процесс автоматического извлечения структурированных данных из текста. Например, это может быть полезно для извлечения IP-адресов, дат, времени, имен и других сущностей [1].
3. Кластеризация текста: это процесс группировки текстовых документов на основе их содержания. Это может быть полезно для выявления скрытых связей между документами и поиска новых угроз безопасности [5].
4. Анализ тональности: это процесс определения тональности текста (например, положительная, отрицательная или нейтральная). Это может быть полезно для анализа общественного мнения и обнаружения угроз в социальных сетях [3].

1.2. Преимущества и ограничения различных методов

Каждый из методов анализа текста имеет свои преимущества и ограничения. Классификация текста может быть эффективна в обнаружении известных угроз, но может не быть эффективна в обнаружении новых угроз. Извлечение информации может быть полезным для сбора информации, но может быть неэффективным в обнаружении скрытых угроз. Кластеризация текста может быть полезной для обнаружения новых угроз, но может быть трудно интерпретировать результаты. Анализ тональности может быть полезным для обнаружения настроений в сообществах, но может быть недостаточным для выявления угроз.

2. Описание алгоритма ChatGPT и его возможностей для выявления угроз безопасности

2.1. Описание основных принципов работы ChatGPT

ChatGPT - это языковая модель на основе трансформера, которая обучается на больших объемах текстовых данных [6]. Она использует механизм самообучения, который позволяет ей генерировать тексты на основе предыдущего контекста. Основным принципом работы ChatGPT является генерация текста на основе заданного контекста. Она работает следующим образом:

1. Обучение модели: ChatGPT обучается на большом объеме текстовых данных, чтобы научиться распознавать и генерировать последовательности слов [6].
2. Генерация текста: после обучения модель использует предыдущий контекст, чтобы сгенерировать следующее слово в последовательности. Она делает это, используя внутренний механизм, который основан на вероятностном распределении слов [1].
3. Обновление модели: когда модель генерирует новый текст, ее параметры могут быть обновлены для улучшения результатов [2].

2.2. Преимущества использования ChatGPT для анализа текстов

ChatGPT обладает рядом преимуществ, которые делают его эффективным инструментом для анализа текстов в области безопасности:

1. Высокая точность: ChatGPT обучается на большом количестве текстов и способен выявлять скрытые зависимости между словами и фразами, что позволяет ему делать точные предсказания [6].
2. Гибкость: ChatGPT может использоваться для решения различных задач, таких как выявление аномалий, обнаружение вредоносного программного обеспечения и анализ текстов на наличие угроз [3, 5, 8].
3. Низкая стоимость: ChatGPT может быть обучен на большом количестве данных с использованием дешевых вычислительных ресурсов, что делает его доступным для широкого круга пользователей [4].

2.3. Описание методов обучения ChatGPT для решения задач безопасности

Одним из главных преимуществ ChatGPT является способность обучаться на небольшом количестве примеров (few-shot learning) [1]. Для этого в модель заранее добавляются некоторые "подсказки" (prompts), которые сообщают ей, какой тип ответа необходимо предсказать. Например, в задаче выявления угроз безопасности в тексте, подсказки

могут быть сформулированы следующим образом: "Обнаружить упоминание о вредоносном ПО в тексте", "Выявить упоминание о хакерских атаках в тексте" и т.д.

Другой важной особенностью ChatGPT является способность к обучению на больших объемах данных. Однако, обучение на больших датасетах может приводить к переобучению модели, поэтому необходимо использовать методы регуляризации, такие как dropout и weight decay [2].

Кроме того, для решения задач безопасности с использованием ChatGPT могут быть применены различные методы аугментации данных, такие как замена синонимов, изменение порядка слов в предложении, удаление и добавление слов [3]. Эти методы позволяют улучшить обобщающую способность модели и снизить вероятность переобучения.

Для защиты от атак на модель, таких как внедрение шума (adversarial noise) и вставка злонамеренных фрагментов текста (adversarial insertions), можно использовать методы адверсариального тренирования (adversarial training) [4]. Они позволяют улучшить устойчивость модели к таким атакам и повысить ее надежность.

В целом, применение ChatGPT для выявления новых угроз безопасности является перспективным направлением и может привести к созданию эффективных систем для защиты информации [3, 5-8].

3. Исследование использования ChatGPT для выявления угроз безопасности

3.1. Описание конкретных задач безопасности, решенных при помощи ChatGPT

ChatGPT имеет широкий спектр применений в области безопасности [5]. Например, она может быть использована для определения потенциальных угроз безопасности, анализа уязвимостей в системах, выявления и предотвращения атак, а также для защиты конфиденциальных данных.

Исследования показывают, что ChatGPT может быть использована для решения задач в области безопасности при помощи обучения с учителем, обучения без учителя и переноса обучения [1, 3, 5–7]. В частности, ChatGPT может использоваться для выявления аномалий в текстовых данных, выявления и классификации угроз безопасности, а также для обнаружения и классификации кибератак.

Одной из конкретных задач, решенных при помощи ChatGPT, является определение подозрительной активности на основе текстовых данных [5]. Для этого ChatGPT обучается на текстовых данных, связанных с безопасностью, например, отчетах о нарушениях безопасности, логах систем и сообщениях о подозрительной активности. Затем ChatGPT может использоваться для автоматического обнаружения подозрительных сообщений и определения категории угрозы безопасности, связанной с сообщением.

Другой пример применения ChatGPT в области безопасности - это определение уязвимостей в системах [3]. ChatGPT может быть обучена на данных об уязвимостях, таких как CVE-сообщения, и использоваться для автоматического определения уязвимостей в новых версиях программного обеспечения.

ChatGPT также может быть использована для предотвращения атак на основе текстовых данных [7]. Например, ChatGPT может использоваться для автоматического обнаружения фишинговых писем и блокировки их отправителей.

Таким образом, ChatGPT представляет собой мощный инструмент в области безопасности, который может быть использован для обнаружения и предотвращения угроз безопасности на основе текстовых данных. Ее применение может значительно повысить безопасность систем и улучшить эффективность работы команд по безопасности [5].

3.2. Анализ результатов эксперимента

В статье "Language models are few-shot learners" было показано, что GPT-3 может быть использован для решения задач, для которых не было предназначено исходных данных обучения. Данные авторы провели эксперимент по созданию и обучению GPT-3 на задаче генерации паролей для защиты пользовательских аккаунтов. Результаты показали, что GPT-3 может генерировать безопасные пароли с высокой степенью точности [1].

В статье "A review of natural language processing techniques in cybersecurity" были представлены различные методы обработки естественного языка (Natural Language Processing, NLP) для решения задач безопасности, в том числе для выявления вредоносных угроз и обнаружения атак. Среди этих методов были описаны методы классификации, кластеризации, анализа тональности, извлечения признаков и моделирования. ChatGPT может быть использован для решения всех этих задач, благодаря своей способности обрабатывать естественный язык и понимать контекст [3].

В работе "A Survey of Adversarial Attacks and Defenses in Natural Language Processing" провели обзор атак и защитных механизмов для NLP, в том числе для моделей глубокого обучения. Авторы подчеркивают, что GPT-3, как и другие языковые модели, подвержен атакам, таким как атаки вставки, замены и удаления символов. Однако существуют методы защиты, которые могут сделать модель более устойчивой к таким атакам [4].

Таким образом, ChatGPT может быть использован для решения конкретных задач безопасности, таких как обнаружение и классификация угроз безопасности на основе текстовых данных, а также для более широкого анализа связей между различными угрозами. ChatGPT может достичь высокой точности в выполнении этих задач.

3.3. Проблемы безопасности при использовании нейросетевой модели GPT

Эксперты по обработке данных опасаются возможных кибератак и утечек важных данных при использовании новой языковой нейросетевой модели GPT от компании OpenAI. Они обнаружили проблемы с безопасностью этой модели, которые могут быть использованы злоумышленниками для создания универсального джейлбрейка, который может обойти большинство языковых моделей.

Универсальный джейлбрейк позволяет злоумышленникам получить доступ к важным данным и проводить кибератаки, в том числе создавать вирусные черви, которые быстро распространяются по всему Интернету. Для борьбы с этой угрозой компании используют различные методы, такие как обучение с подкреплением и обратную связь от человека. Однако, чтобы решить эту проблему безопасности, требуется особое внимание и работа над ее решением.

Некоторые компании уже работают над системами, которые могут обнаруживать джейлбрейки и атаки с внедрением подсказок автоматически с помощью других моделей. Кроме того, существуют дополнительные пути решения проблемы, такие как использование второй модели для анализа подсказок первой и отделение подсказки системы от подсказки пользователя.

В целом, проблема безопасности языковых моделей является серьезной и требует постоянного внимания и работы над ее решением, чтобы обеспечить безопасность в Интернете. Компании, такие как OpenAI, Google и Microsoft, продолжают работать над усовершенствованием своих моделей и борьбой с этой проблемой, однако, существует необходимость в постоянном обновлении и усовершенствовании методов защиты, чтобы минимизировать риски для пользователей.

Заключение

В заключении были сделаны выводы о применимости ChatGPT для выявления угроз безопасности, подчеркнуто его эффективность в анализе текстов и способность обучаться на небольшом количестве данных. Однако были также указаны ограничения ChatGPT, связанные с необходимостью большого количества вычислительных ресурсов и возможными проблемами при анализе специализированной терминологии или сленга.

Были обсуждены возможные направления дальнейших исследований, такие как улучшение точности и надежности ChatGPT при анализе больших объемов данных, а также его применение для решения более сложных задач в области кибербезопасности.

Несмотря на ограничения, ChatGPT представляет перспективное направление в области выявления угроз безопасности. Его способность обучаться на небольшом количестве данных и выявлять новые угрозы делает его особенно полезным для применения в кибербезопасности. В дальнейшем исследовании стоит уделить внимание разработке более эффективных алгоритмов обучения ChatGPT, а также рассмотреть возможность использования других языковых моделей для выявления угроз безопасности.

При всевозможных плюсах использования ChatGPT, эксперты по обработке данных предупреждают об опасности кибератак и утечек данных при использовании новой языковой нейросетевой модели. Проблемы с безопасностью могут быть использованы злоумышленниками для создания универсального джейлбрейка, который может обойти большинство языковых моделей. Проблема безопасности языковых моделей остается серьезной и требует постоянного внимания и улучшения методов защиты, чтобы минимизировать риски для пользователей.

Литература

1. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.
2. Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2019). On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836.
3. Nguyen, H., Nguyen, C., Nguyen, M., Nguyen, H., Nguyen, L., & Dao, M. (2020). A review of natural language processing techniques in cybersecurity. arXiv preprint arXiv:2012.01254.
4. Zhang, R., Li, T., Wang, B., & Liu, B. (2021). A Survey of Adversarial Attacks and Defenses in Natural Language Processing. arXiv preprint arXiv:2102.09605.
5. Yuan, J., Zhang, H., Xu, L., & Ren, K. (2021). Natural language processing in cybersecurity: A review. Journal of Network and Computer Applications, 182, 103038.
6. Lin, Y., Shen, S., Liu, Z., Luan, H., & Sun, M. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv preprint arXiv:1910.10683.
7. Garg, S., Ghosh, S., & Singh, A. (2021). Exploring Adversarial Attacks and Defenses in Natural Language Processing. arXiv preprint arXiv:2103.03155.
8. Guo, H., Wang, Y., & Liu, T. (2020). Language Models for Attack and Defense in Natural Language Processing: A Survey. arXiv preprint arXiv:2012.15779.

УДК 004

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ЗАДАЧИ РАСПРЕДЕЛЕНИЯ НАГРУЗКИ ПРЕПОДАВАТЕЛЕЙ НА КАФЕДРЕ

Д. А. Крамаренко

Воронежский государственный университет

Введение

Настоящая работа посвящена разработке математической модели и базы данных задачи распределения нагрузки преподавателей на кафедре.

Задача распределения кафедральной нагрузки между преподавателями решается ежегодно и требует достаточно много усилий в силу изменяющихся внешних условий. В частности, на кафедрах может изменяться состав преподавателей, поэтому встает задача перераспределения имеющейся нагрузки или поиска квалифицированных кадров для ведения дисциплин. Кроме того, часто меняются учебные планы: часть дисциплин убирается, вместо них добавляются другие.

Несмотря на изменяющиеся внешние условия, образовательный процесс должен проходить без срывов, дисциплины должны преподаваться качественно компетентными преподавателями. Без использования специализированного программного обеспечения, основанного на подходящей математической модели, задача распределения нагрузки становится достаточно сложной, особенно в случае, когда кафедра многочисленна и объем нагрузки большой. Поэтому встает задача разработки соответствующей математической модели и реализованного на ее основе прикладного решения.

1. Математическая модель

Для построения математической модели определим исходные данные и начальные условия. Нагрузка кафедры представляет собой общее количество часов, разбитое на дисциплины в соответствии с учебными планами, с указанием курса, потока, групп, в которых должны проводиться занятия по этой дисциплине.

Преподаватели кафедры имеют каждый свою квалификацию и специализируются в определенном наборе дисциплин. При этом базовые дисциплины кафедры, как правило, могут вести почти все преподаватели, а специальные дисциплины очень часто может вести только один преподаватель кафедры. В связи с этим необходимо определить для каждого преподавателя те дисциплины, которые он может вести в силу своей квалификации.

Пусть $P = \{p_1, p_2, \ldots, p_n\}$ множество преподавателей кафедры, $S = \{s_1, s_2, \ldots, s_m\}$ – множество элементов учебной нагрузки. Элемент учебной нагрузки s_i представляет собой наименование дисциплины с указанием формы проведения занятий (лекция, лабораторная работа, аудиторная работа и т.д.), группы, подгруппы или потока, в которых проводятся занятия.

Для каждого преподавателя $p_i \in P$ определим 3 множества:

1) Дисциплины, которые преподаватель может вести, исходя из своей квалификации и занимаемой должности $C(p_i) \subseteq S$.

2) Дисциплины, которые «исторически» закрепились за преподавателем $D(p_i) \subseteq C(p_i)$.

3) Дисциплины, которые преподаватель хотел бы вести $B(p_i) \subseteq S$.

Каждому элементу нагрузки $s_j \in S$ ставится в соответствие количество часов $w(s_j)$ общей нагрузки кафедры *W* на учебный год, т.е:

$$\sum_{i=1}^{m} w(s_j) = W \tag{1}$$

Каждому преподавателю p_i ставится в соответствие количество часов нагрузки по доле ставки $W_{cm}(p_i)$.

Нагрузка преподавателя p_i представляет собой подмножество множества дисциплин учебной нагрузки, которые преподаватель может вести, $S(p_i) \subseteq C(p_i)$ количество часов нагрузки преподавателя равно

$$W(p_i) = \sum_{j:s_j \in S(p_i)} w(s_j), \tag{2}$$

Количество часов нагрузки преподавателя $W(p_i)$ должно соответствовать количеству часов по доле ставки $W_{cm}(p_i)$. Поскольку точное равенство на практике невыполнимо, введем параметр δ – допустимое отклонение фактического количества часов от количества часов по ставке. Тогда:

$$\left|W(p_i) - W_{cm}(p_i)\right| \le \delta, i = 1, \ldots, n \tag{3}$$

Так как для каждого преподавателя заданы множества $C(p_i)$, $D(p_i)$ и $B(p_i)$, то каждая из дисциплин нагрузки имеет определенную ценность для преподавателя. При этом дисциплина из множества $D(p_i)$ и $B(p_i)$ имеет большую ценность, чем дисциплины из множества $C(p_i)$.

Определим величину c_{ij} - «ценность» дисциплины s_i для преподавателя p_j. Тогда логично вывести критерий:

$$\sum_{i=1}^{n} \sum_{j:s_j \in S(p_i)} c_{ij} \to \max, \tag{4}$$

Введем переменные:

$$x_{ij} \begin{cases} 1, \text{если дисциплина } s_i \text{ закреплена за преподавателем } p_i \\ 0 \text{ в противном случае} \end{cases} \tag{5}$$

Тогда приходим к следующей оптимизационной задаче:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \to \max, \tag{6}$$

$$\bigcup_{i=1}^{n} S(p_i) = S, \tag{7}$$

$$S(p_i) \subseteq C(p_i), \tag{8}$$

где $S(p_i) = \{s_j : x_{ij} = 1\}$,

$$\left|W\left(p_i\right)-W_{cm}\left(p_i\right)\right| \le \delta, i=1,\ldots,n, \quad (9)$$

где $W(p_i)=\sum_{j:s_j \in S(p_i)} w(s_j)$,

$$\sum_{i=1}^{n} W(p_i) = W, \quad (10)$$

$$x_{ij} = \{0,1\}. \quad (11)$$

2. База данных

Для решения поставленной задачи было решено разработать конфигурацию на базе платформы «1С: Предприятие 8.3». Конфигурация поможет определить структуру базы данных. Реквизиты и табличные части объектов конфигурации «Справочник» должны хранить всю необходимую информацию для обеспечения корректной работы прикладного решения.

Справочник «Преподаватели» должен содержать основную информацию о преподавательском составе, такую как: фамилия, имя, отчество, должность (профессор, доцент, преподаватель) и занятость (штатный, совместимый). Для удобства хранения значений должностей и занятостей были созданы соответствующие объекты конфигурации «Перечисления». Для хранения информации об ученых степенях и званиях преподавателя следует также создать дополнительные справочники «Ученые степени» и «Ученые звания».

У каждого преподавателя имеются множества дисциплин, которые он может вести исходя из своей квалификации, которые закрепились за ним «исторически» и которые он изъявляет желание вести. Информация об этих дисциплинах содержится в табличных частях справочника, представляющих собой списки с необходимыми предметами.

Справочник «Дисциплины» должен хранить характеристику дисциплины, в особенности количество часов, выделенных на лекции, практические занятия и т.д. Указывается кафедра, к которой прикреплена дисциплина, список групп в которой дисциплина преподается (в виде табличной части), а также семестр, для которого создано соответствующее перечисление.

Кафедры, как и группы студентов, необходимо также представить в соответствующих справочниках.

Схема полученной базы данных представлена на рис. 1.

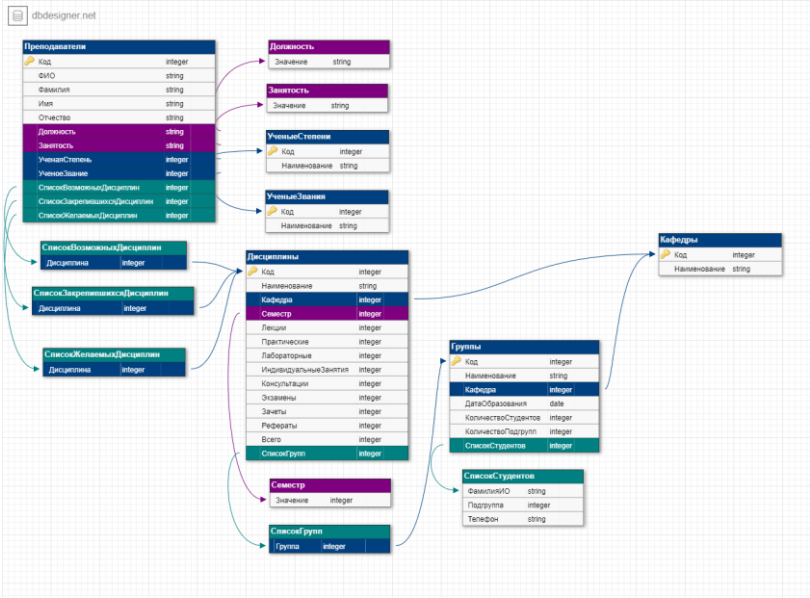


Рис. 1. Схема базы данных

Поля, являющиеся ссылочными объектами, выделены цветом. Зеленые таблицы, показанные на схеме, представляют табличные части справочников, и также имеют свои реквизиты, хранящие необходимые данные.

Заключение

В данной статье сформулирована математическая модель задачи распределения учебной нагрузки преподавателей на кафедре. Описана структура спроектированной базы данных для решения поставленной задачи. Оптимизация распределения учебной нагрузки обеспечит более продуктивную учебную работу кафедры.

Литература

1. Шикульский М. И. Математическая модель и алгоритм распределения и контроля учебной нагрузки между профессорско-преподавательским составом / М. И. Шикульский, Е. М. Евсина, Е. П. Кравченкова // Инженерно-строительный вестник Прикаспия. – 2022.– № 3.– С. 151–157.

2. Шахова Е. Ю. Моделирование распределения рабочего времени преподавателей / Е. Ю. Шахова // Статистика и экономика. – 2017. – № 1. – С. 11–23.

3. Султанова С. Н. Модели и алгоритмы поддержки принятия решений при распределении учебной нагрузки преподавателей / С. Н. Султанова, С. В. Тархов // Вестник Уфимского государственного авиационного технического университета. – 2006. – Т. 7. – № 3 (16). – С. 107–114.

4. Радченко М. Г. и работа с данными «1С:Предприятия 8.2» / М. Г. Радченко, Е. Ю. Хрусталева. – Москва : ООО «1С-Паблишинг», 2011. – 268 с.

УДК 519.85

ЗАДАЧА КОММИВОЯЖЁРА И МЕТОДЫ ЕЁ РЕШЕНИЯ

К. А. Кретова, Е. М. Аристова

Воронежский государственный университет

Введение

Задача коммивояжёра – одна из самых известных задач комбинаторной оптимизации. Её суть заключается в поиске самого выгодного маршрута, проходящего через указанные города по одному разу с последующим возвратом в исходный город [1].

Данная статья посвящена исследованию этой задачи и различным методам её решения на примере поиска кратчайшего прогулочного маршрута для студента, который хочет посетить красивые и интересные места Воронежа.

1. Теоретическая часть

Имеется n городов. Задана матрица расстояний между ними: $C = (c_{ij}),\ i,j = \overline{1,n}$. Будем считать, что $c_{ij} \geq 0,\ \forall i,j$. Пусть необходимо найти кратчайший замкнутый маршрут, проходящий через каждый город ровно один раз и минимизирующий суммарное пройденное расстояние.

Математическая постановка задачи может быть записана, например, таким образом:

$$L = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to min,$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = \overline{1,n},$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = \overline{1,n},$$

$$x_{ij} = \{0;1\},\ i,j = \overline{1,n}.$$

При этом $x_{ij} = 1$, если путешественник переезжает из i-ого города в j-ый и $x_{ij} = 0$, если j-ый город не посещается.

Формально введём $(n+1)$ город, расположенный там же, где и первый город, т.е. расстояния от $(n+1)$ города до любого другого, отличного от первого, равны расстояниям от первого города. При этом, если из первого города можно лишь выйти, то в $(n+1)$ город можно лишь прийти.

Введём дополнительные целые переменные, равные номеру посещения этого города на пути, $u_1 = 0$, $u_{n+1} = n$. Тогда дополнительное требование отсутствия подциклов может быть записано аналитически так [2,3]:

$$u_i - u_j + nx_{ij} \leq n-1, \quad j = 2,\ldots,n+1, \quad i = 1,\ldots,n, \quad i \neq j, \quad \text{при } i = 1,\ j \neq n+1$$

$$0 \leq u_i \leq n, \quad x_{in+1} = x_{i1}, \quad i = 2,\ldots,n.$$

2. Постановка задачи

Рассмотрим конкретную задачу. Пусть имеется пять точек на карте Воронежа, которые собирается посетить студент в ходе пешей прогулки:

- вершина 1 – Главный корпус ВГУ (пл. Университетская, д.1) – начальная точка, из которой студент должен начать свой путь;
- вершина 2 – Кольцовский сквер (ул. Дзержинского, 12А) – зелёный парк в самом центре города со свето-музыкальным фонтаном, бюстом поэта А.В.Кольцова – одним из старейших памятников в Воронеже, а также рядом экзотических растений на территории;
- вершина 3 – Советская площадь (ул. Советская, 1в) – одна из старейших площадей города, на которой расположен Воронежский концертный зал и сухой музыкальный фонтан;
- вершина 4 – Адмиралтейская площадь (Петровская набережная) – площадь, посвящённая строительству флота в Воронеже во времена Петра I. К набережной также пришвартован корабль-музей «Гото Предестинация»;
- вершина 5 – Петровский сквер (ул. Степана Разина) – небольшой сквер с памятником Петру I и фонтаном, который является своего рода визитной карточкой города.

При этом в конце маршрут должен заканчиваться там же, где и начался, – в Главном корпусе ВГУ (вершина 1).

Матрица расстояний приведена в табл. 1 [4]:

Таблица 1

Матрица расстояний между вершинами (в метрах)

	1	**2**	**3**	**4**	**5**
1	–	700	1200	1000	2200
2	700	–	800	1400	1500
3	1200	800	–	1200	1300
4	1000	1400	1200	–	2400
5	2200	1500	1300	2400	–

Прочерки на главной диагонали означают, что такие пути не допустимы, так как их введение в маршрут даст наличие подцикла. Это обусловлено главным ограничением задачи коммивояжёра – запретом посещать любую вершину более одного раза.

3. Способы решения

Можно выделить три подгруппы методов решения задачи коммивояжёра:

1) *точные методы*, которые не только находят некоторое решение, но и при окончании своей работы доказывают, что это решение – лучшее. К таким методам относятся:

- полный перебор – работает быстро только для небольших значений n;
- направленный поиск с возвратами – перебор вариантов «вокруг» некоторого решения с отсечением путей, имеющих длину большую, чем лучший к текущему моменту путь;
- метод ветвей и границ – наиболее эффективный из известных метод отсечения «неперспективных» узлов за счёт анализа матрицы расстояний. В этом методе при поиске оптимального решения строится бинарное дерево [5–7];

2) *эвристические методы* – методы, сокращающие полный перебор, но при этом не гарантирующие получения точного решения и скорее дающие приближённое решение. Такими методами, например, являются:

- жадный алгоритм – метод ближайшего соседа;
- метод шнурка – геометрическая вариация жадного алгоритма. Суть данного метода заключается в следующем: сперва все города охватываются замкнутым контуром, который постепенно растягивается, стараясь пройти через все города, минимально увеличив при этом свою длину;
- скользящий перебор – является способом улучшения уже найденного неточного решения и заключается в перестановке соседних городов в пути с целью нахождения наименьшего по длине варианта [7,8];

3) *вероятностные методы*, которые фактически никогда не останавливаются, совершая случайные изменения пути, в ожидании получения более короткого. К таким методам, например, относятся:

- метод отжига – метод, являющийся модификацией вероятностного метода градиентного спуска, суть которого заключается в перестановке городов с постепенно затухающей интенсивностью и сохранением наилучшего из найденных решений;
- генетический алгоритм – усложнённый вариант предыдущего метода, в котором найденные пути постоянно «мутируют» и «скрещиваются» друг с другом, обмениваясь своими отдельными участками [7,9].

В данной работе для решения задачи коммивояжёра будут рассмотрены два метода: один точный метод решения – полный перебор городов и один эвристический – метод ближайшего соседа.

3.1. Полный перебор

Так как в имеющейся задаче значение n маленькое, то самый простой и удобный способ решения – полный перебор вариантов путей с нахождением кратчайшего.

Для данного варианта получаем следующее решение, представленное в табл. 2.

Таблица 2

Перебор путей

Путь	**Сумма расстояний**	**Итоговое расстояние**
1 → 2 → 3 → 4 → 5 → 1	700 + 800 + 1200 + 2400 + 2200	7300
1 → 2 → 3 → 5 → 4 → 1	700 + 800 + 1300 + 2400 + 1000	6200
1 → 2 → 4 → 3 → 5 → 1	700 + 1400 + 1200 +1300 + 2200	6800
1 → 2 → 4 → 5 → 3 → 1	700 + 1400 + 2400 + 1300 + 1200	7000
1 → 2 → 5 → 3 → 4 → 1	700 + 1500 + 1300 + 1200 + 1000	5700
И т.д.		

Продолжая перебирать варианты, получаем, что самый оптимальный маршрут пешей прогулки является следующим: Главный корпус ВГУ → Кольцовский сквер → Петровский сквер → Советская площадь → Адмиралтейская площадь → Главный корпус ВГУ. Его длина составит 5,7 км.

3.2. Метод ближайшего соседа

Алгоритм ближайшего соседа относится к категории «жадных» алгоритмов и так же является одним из простейших методов решения задачи коммивояжёра.

Данный алгоритм быстро выполняется, прост в реализации, но, как и другие эвристические методы решения, может выдавать неоптимальные результаты.

Шаги алгоритма:

1. Обозначить все вершины как не посещённые;
2. Выбрать начальную вершину *v* и пометить её, как посещённую;
3. Выбрать наиближайшую не посещённую смежную вершину *u* к вершине *v*;
4. Обозначить *u* как текущую вершину и пометить её как посещённую;
5. Если все вершины помечены как посещённые, то завершить алгоритм. Иначе – вернуться к шагу 3.

На выходе будем иметь последовательность вершин предположительно оптимального решения [10,11].

Решение приведённой выше задачи данным способом приведено в табл. 3:

Таблица 3

Решение задачи методом ближайшего соседа

Текущая вершина	**Множество не посещённых вершин**	**Множество посещённых вершин**	**Ближайшая вершина**	**Расстояние до ближайшей вершины**	**Итоговое расстояние**
1	{2, 3, 4, 5}	{1}	2	700	700
2	{3, 4, 5}	{1, 2}	3	800	1500
3	{4, 5}	{1, 2, 3}	4	1200	2700
4	{5}	{1, 2, 3, 4}	5	2400	5100
5	Ø	{1, 2, 3, 4, 5}	1	2200	7300

В данном случае получаем следующий оптимальный маршрут прогулки: Главный корпус ВГУ → Кольцовский сквер → Советская площадь → Адмиралтейская площадь → Петровский сквер → Главный корпус ВГУ. Его длина составляет 7,3 км.

Заключение

В данной статье была рассмотрена задача о поиске оптимального пути с отсутствием подциклов – задача коммивояжёра. Так же была приведена её математическая модель и подробно изучены методы её решения, сформулирована и решена двумя способами задача о поиске оптимального прогулочного маршрута по Воронежу в определённом районе города. Вычислительный эксперимент подтвердил, что алгоритм ближайшего соседа при решении задачи коммивояжёра, как и любой другой «жадный» алгоритм, может выдавать неоптимальные решения, в отличии от точного метода – полного перебора всех возможных маршрутов.

Доцент кафедры вычислительной математики и прикладных информационных технологий факультета прикладной математики, информатики и механики Воронежского государственного университета, кандидат физико-математических наук, Аристова Екатерина Михайловна.

Литература

1. Задача коммивояжёра. – Режим доступа: https://ru.wikipedia.org/wiki/Задача_коммивояжёра. – (Дата обращения: 20.02.2023).
2. Булгакова, И.Н. Модели и методы дискретной оптимизации: учебное пособие / И.Н. Булгакова, Г.Д. Чернышова. – Воронеж: Издательский дом ВГУ, 2021. – 76 с.
3. Математическая модель задачи коммивояжёра. – Режим доступа: https://math.semestr.ru/kom/index.php. – (Дата обращения: 20.02.2023).
4. Google Карты. https://www.google.com/maps. – (Дата обращения: 20.02.2023).
5. Метод ветвей и границ. Задача коммивояжёра. – Режим доступа: https://habr.com/ru/post/560468. – (Дата обращения: 01.03.2023).
6. Travelling salesman problem. – Режим доступа: https://en.wikipedia.org/wiki/Travelling_salesman_problem. – (Дата обращения: 01.03.2023).
7. Задача коммивояжёра. – Режим доступа: https://synset.com/ai/ru/tsp/Salesman_Intro.html. – (Дата обращения: 01.03.2023).
8. Эвристики в задаче коммивояжёра – Режим доступа: https://synset.com/ai/ru/tsp/Salesman_Heuristics.html. – (Дата обращения: 01.03.2023).
9. Задача коммивояжёра. Метод имитации отжига. – Режим доступа: http://mech.math.msu.su/~shvetz/54/inf/perl-examples/PerlExamples_CommisVoyageur.xhtml#PerlExamples_CommisVoyageur_Commons_OptimizationProblems. – (Дата обращения: 01.03.2023).
10. Алгоритм ближайшего соседа в задаче коммивояжёра. – Режим доступа: https://ru.wikipedia.org/wiki/Алгоритм_ближайшего_соседа_в_задаче_коммивояжёра. – (Дата обращения: 06.03.2023).
11. Задача коммивояжёра. – Режим доступа: https://studfile.net/preview/7050839/page:28. – (Дата обращения: 06.03.2023).

УДК 004.94

МОДЕЛИРОВАНИЕ СИМУЛЯЦИИ ТРАВЫ НА ИГРОВОМ ДВИЖКЕ UNITY3D

А.С. Крутько, Е.В. Трофименко

Воронежский государственный университет

Введение

В последние годы появляется все больше новых возможностей сделать игровые и компьютерные миры более насыщенными и реалистичными. Одним из ключевых факторов, влияющих на восприятие пользователем виртуальных миров, является проработка мелких деталей, таких как растительность. В частности, трава является важным элементом для создания реалистичного и живого окружения. Моделирование симуляции травы на игровом движке Unity3D становится все более популярным среди разработчиков игр и виртуальной реальности.

В настоящей статье рассматривается процесс моделирования симуляции травы на игровом движке Unity3D с использованием шейдеров и тесселяции [1, 2].

1. Описание метода

Одним из основных подходов к симуляции травы является использование геометрических шейдеров с тесселяцией. Данный подход позволяет за наименьшее количество вызовов отрисовки (draw call) получить желаемый результат в виде объекта травы. Для придания картинке динамичности полученный объект травы анимируется с помощью текстуры шумов.

1.1 Тесселяция, геометрический шейдер

Шейдер – компьютерная программа, предназначенная для исполнения процессором видеокарты. При работе с трехмерной графикой программы используют шейдеры для определения параметров геометрических объектов или эффектов, наложенных на данные объекты. В работе будет использован как обычный шейдер, так и геометрический, что позволит формировать травинки из примитивов, а после управлять их цветом, отражением света и тенями. Кроме использования шейдеров, в работе также используется тесселяция – автоматизированный процесс добавления новых выпуклых многоугольников в полигональную сетку. Использование тесселяции в паре с геометрическим шейдером позволит увеличить количество отображаемых травинок и создать полноценный объект в виде травы.

1.2 Шум Перлина

Шум Перлина часто находит свое применение в компьютерной графике [3], с его помощью моделируют облака, генерируют уровни игрового пространства и многое другое. В данной работе текстура шума Перлина используется для создания искусственного влияния

ветра на траву. Данный шум имеет простую генерацию, которая позволяет получить разнообразные изображения шумов. Полученные изображения накладываются на объект травы, затем, параметр, отвечающий за наклон травинки, умножается на значение цвета в координатах изображения шума, которые соответствуют координатам данной травинки. В результате проведенных вычислений полученное значение используется в качестве параметра, отвечающего за наклон травинки под воздействием ветра.

2. Реализация

При симуляции работы травы в игровом движке Unity3D необходимо реализовать следующие этапы:

1. Необходимо написать код для шейдеров, которые будут отвечать за симуляцию травы. Для реализации этого используется стандартный шейдер, отвечающий за настройку отображения объекта, и геометрический шейдер, отвечающий за отрисовку травинок с использованием примитивов. В них также необходимо реализовать:
   - настройку высоты и ширины травинок, их положения и количества;
   - настройку «ветра» - её покачивания, направления и шаблона поведения на ветру;
2. Также необходимо реализовать набор инструментов редактора, которые упростят использование шейдера разработчиком, таких как:
   - инструмент по редактированию свойств шейдера у объектов;
   - инструмент по генерации новых текстур шумов для создания разнообразных поведений травы на ветру;
3. Чтобы показать пример работы необходимо создать базовый объект – «префаб», а также сцену в которой будет показана работа с шейдером.

Задача была реализована с использованием языка программирования C#, HLSL [4] и ShaderLab [5] для создания шейдеров.

При создании данного набора инструментов (далее - ассетов) для игрового движка Unity3D была разработана следующая диаграмма вариантов использования, представленная на рисунке 1.

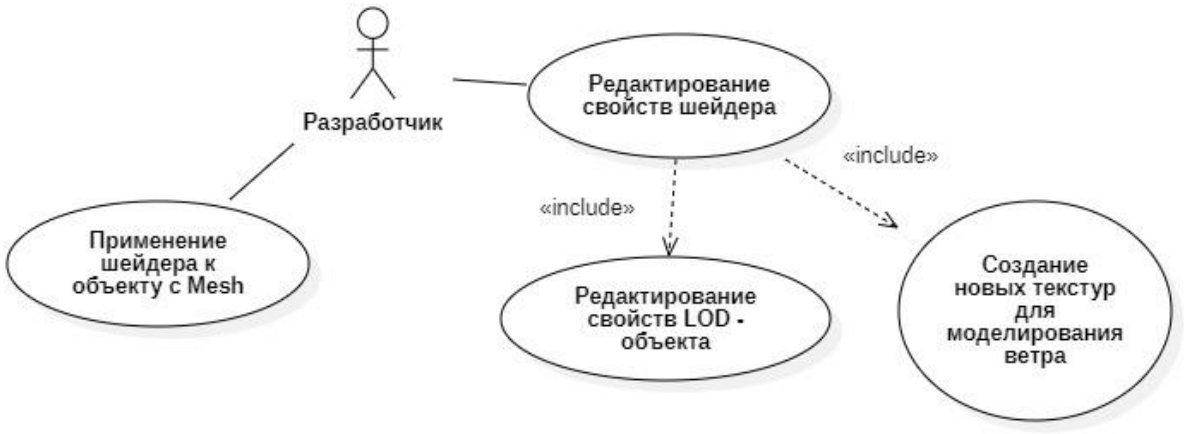


Рис. 2. Диаграмма вариантов использования проекта

На данной диаграмме обозначены все необходимые варианты использования ассетов, которые могут пригодится разработчику для реализации большинства используемых функций.

Основной принцип, по которому будет работать данный ассет, это использование геометрического шейдера и тесселяции. Каждый из этих этапов рендера не является

обязательным и если в случае с геометрическим шейдером в Unity3D есть своя имплементация данного шейдера, которую можно использовать «из коробки», то в случае же с тесселяцией необходимо определять свой поверхностный шейдер (hull) и доменный шейдер (domain). Пример конвейера рендера с этапами поверхностного шейдера, тесселяции и доменного шейдера представлены на рисунке 2. Каждый из данных этапов можно описать следующим образом:

1. Поверхностный шейдер отвечает за подготовку и передачу данных для обработки их на этапе работы тесселяции и доменного шейдера.
2. Тесселяция отвечает за генерацию набора объектов меньшего размера из полученной геометрии.
3. Доменный шейдер отвечает за вычисление положения вершины разделенной точки и UV-координат, относящихся к ней.

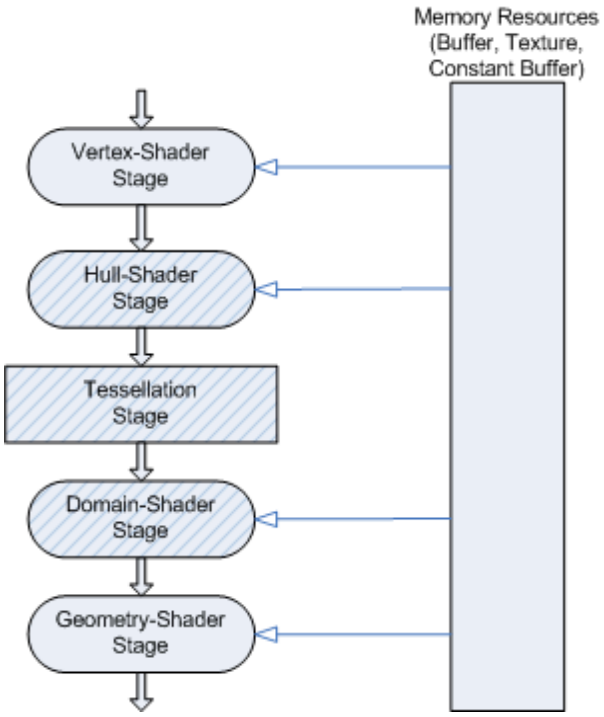


Рис. 3. Диаграмма этапов графического конвейера Direct3D

Как и в случае с OpenGL – конвейер рендера Unity3D производит этап тесселяции перед работой геометрического шейдера, таким образом, разделив полигон на подмножество более маленьких полигонов, Unity3D использует полученные вершины для создания примитивов. Схема алгоритма, согласно которому производится обработка данных в процессе работы шейдера представлен на рисунке 3.

Рис. 4. Схема основного алгоритма работы геометрического шейдера

Каждый из этапов алгоритма можно описать следующим образом:

1. Этап подсчёта основных характеристик травинки. На данном этапе вычисляется следующее:
    - матрица перевода координат из системы координат касательных в систему координат локальных;
    - матрица, определяющая направление, в которое должны «смотреть» травинки;
    - матрица, определяющая поворот сегмента травинки для создания эффекта изгиба.
2. Этап подсчёта значений, определяющих каким образом ветер, должен влиять на траву: используя позицию травинки, высчитывается UV координата для текстуры (изображения) шума, с помощью полученной координаты высчитывается и нормализуется вектор, который далее используется для получения значения поворота, задаваемого ветром.
3. На этапе сбора травинки из сегментов проводятся следующие вычисления:
    - из полученных ранее значений формируются матрица трансформации для самого нижнего сегмента травинки и матрица трансформации для остальных сегментов травинки;
    - в цикле, используя полученные матрицы трансформаций, информация записывается в поток вершин, структура которую хранит каждая из данных вершин представлена на рисунке 4.

```
// Structs
struct geometry_output
{
    // Позиция вершины
    float4 pos : SV_POSITION;
    // UV координата вершины
    float2 uv : TEXCOORD0;
    // Текстурная координата вершины
    unityShadowCoord4 _ShadowCoord : TEXCOORD1;
    // Нормаль, относящяяся к данной вершине
    float3 normal : NORMAL;
};
```

Рис. 5. Структура, которую используют вершины для хранения данных

В результате работы описанного выше алгоритма получается сцена, представленная на рисунке 5.

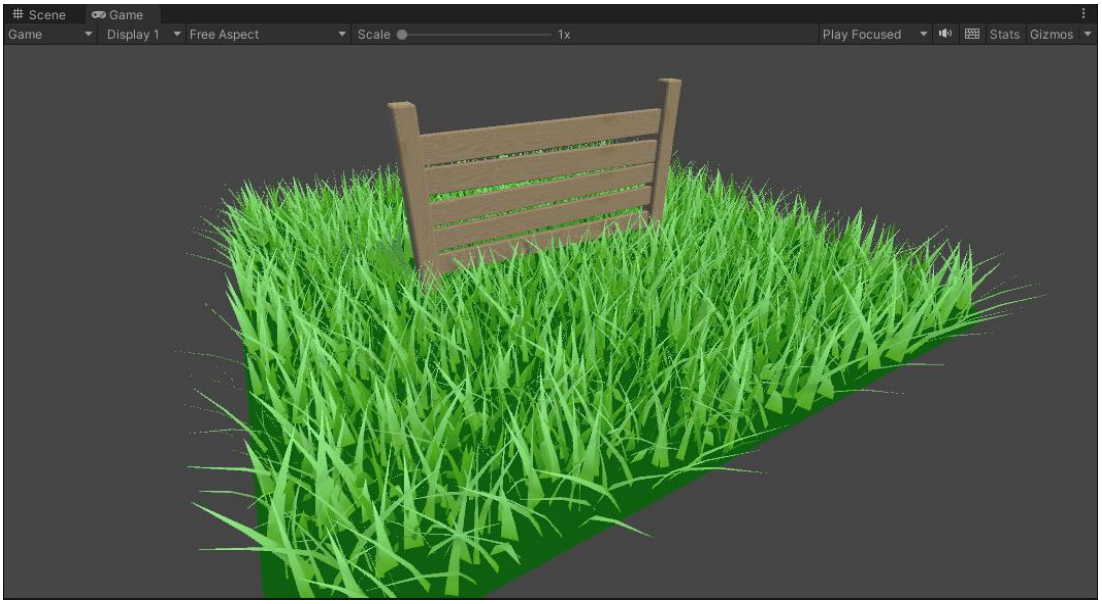
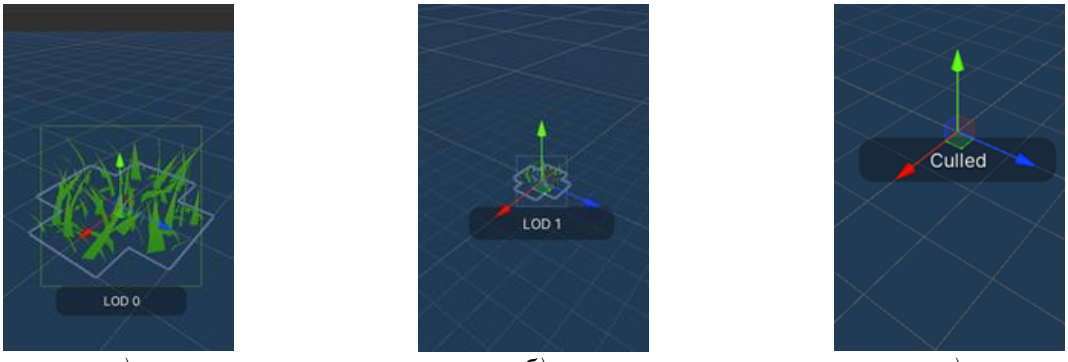


Рис. 6. Результат работы шейдера

В игровом движке Unity3D есть возможность использования LOD (level of detail) – инструмента, используемого переключения между уровнями детализации для объекта. В случае данной работы для снижения детализации можно уменьшать количество травинок, которые генерируются с помощью тесселяции. Следовательно, настраивается группа LOD объектов для 3 состояний следующим образом:

1. Генерация на объекте префаба необходимого количества травинок с помощью тесселяции.
2. Генерация на объекте префаба в два раза меньшего количества травинок (можно настроить через параметры шейдера).
3. Использование «culled» состояния объекта, при котором он не отображается при рендере.

Каждое из трех описанных выше состояний можно использовать для разных расстояний до объекта префаба, чтобы уменьшить нагрузку на систему и оптимизировать работу программы. Результат работы настроенного префаба с LOD показан на рисунке 6.



а) *б)* *в)*

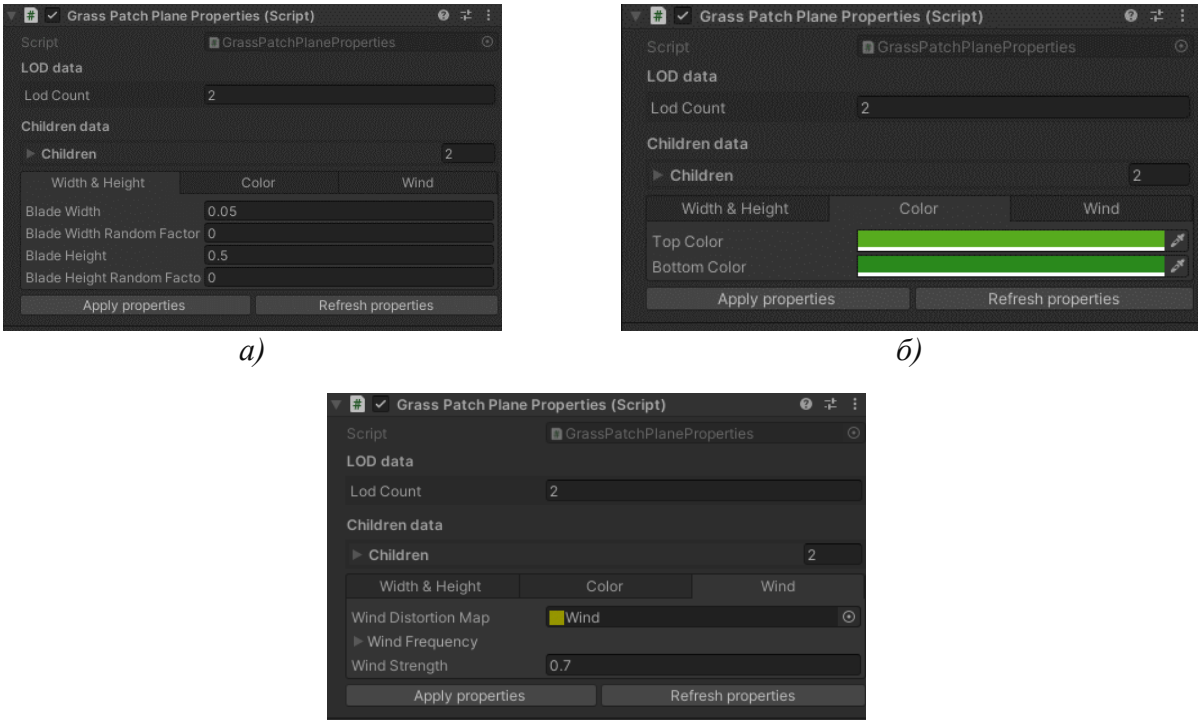
Рис. 7. Префаб с настроенным LOD

а) камера максимально приближена к объекту - отрисовывается максимально возможное количество травинок;
б) камера находится на «среднем» расстоянии от объекта - отрисовывается в два раза меньше травинок чем в пункте а рисунка 6;

в) камера находится на большом расстоянии от объекта - объект переводится culled режим и не отрисовывается на экране пользователя.

В рамках данной работы для облегчения управления параметрами шейдера был создан пользовательский редактор свойств шейдера для объектов на сцене, представленный на рисунке 7, где в каждом из пунктов открыта определенная вкладка редактора:

а) параметры высоты и ширины травы;
б) параметры цвета травы;
в) параметры ветра.



а) *б)*

в)

Рис. 8 Редактор свойств шейдера в LOD объектах

Кроме инструмента для работы со свойствами шейдера также был создан инструмент, позволяющий генерировать текстуры (изображения) шума для использования их в шейдере, чтобы устанавливать определенное влияние ветра на траву. Окно данного инструмента представлено на рисунке 8.

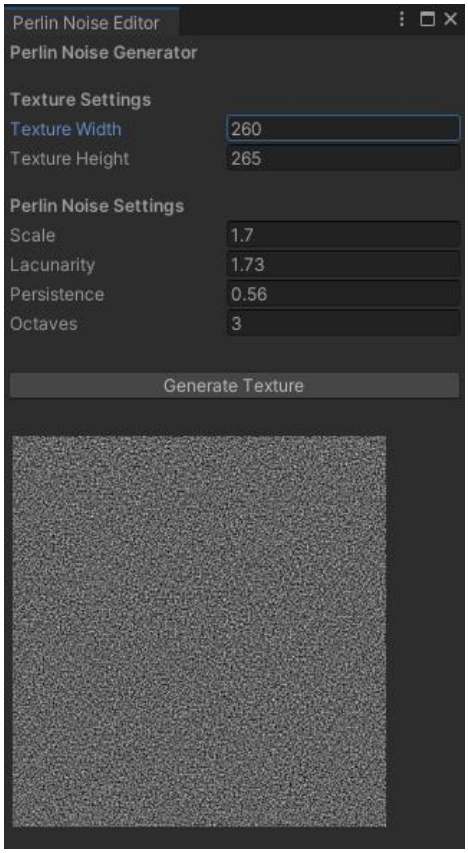


Рис. 9. Генератор шума Перлина

Данный инструмент позволит разработчику создавать новые шумы, с помощью которых будет задаваться поведение травы на ветру.

Заключение

В данной статье была рассмотрена задача создания реалистичной симуляции травы на игровом движке Unity3D. Был рассмотрен метод лежащий в основе подобного рода симуляций. Также были предложены инструменты для упрощения работы разработчика с шейдером. В результате был создан префаб и сцена для демонстрации работы разработанного ассета.

Литература

1. Surface Shaders with DX11 / OpenGL Core Tessellation. Unity Documentation. – Режим доступа: https://docs.unity3d.com/Manual/SL-SurfaceShaderTessellation.html (Дата обращения: 10.04.2023).
2. Этапы тесселяции. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/direct3d11/direct3d-11-advanced-stages-tessellation?redirectedfrom=MSDN (Дата обращения: 10.04.2023).
3. Гинсбург Д. OpenGL ES 3.0. Руководство разработчика. /Д Гинсбург, Б. Пурномо; М:ДМК-Пресс, 2015 г. — 449 с.
4. HLSL in Unity. Unity Documentation. – Режим доступа: https://docs.unity3d.com/Manual/SL-ShaderPrograms.html (Дата обращения: 10.04.2023).
5. ShaderLab. Unity Documentation. – Режим доступа: https://docs.unity3d.com/Manual/SL-Reference.html (Дата обращения: 10.04.2023).

УДК: 004.9

АРХИТЕКТУРА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ДЛЯ ОБУЧЕНИЯ НЕЧЕТКОЙ ЛОГИКЕ

Н.М. Кушнарёв, Е.М. Аристова

Воронежский государственный университет

Введение

Обучение чему-либо, бесспорно, остается одной из важных частей человеческого развития. Развитие технологий позволяет реализовывать сложные системы, облегчающие процесс обучения. Такие системы могут предлагать собственные программы обучения или создавать отдельные блоки заданий, чтобы заказчики могли выделить больше ресурсов для расширения разнообразия создаваемого контента.

Генерация заданий решает проблему недостатка практики пользователей по той или иной теме, дает возможность разработчикам сконцентрировать внимание на качестве, а вопрос количества перекладывается на систему. Задания могут быть различной сложности, заложенной экспертами в определенной области. Таким образом, задача реализации генерации заданий заключается в написании алгоритмов, учитывающих предметную область решаемой задачи и ее особенности и заложенную сложность. Такие алгоритмы учитывают от каких характеристик зависит сложность задания и как она регулируется.

1. Общая структура разработанного приложения

1.1. Общее описание приложения

В ходе изучения данной проблемной области было разработано обучающее клиент-серверное приложение, реализующее генерацию заданий по нечеткой логике. Ее архитектура позволяет расширять приложение новым контентом, заданиями и новыми темами для обучения, при этом добавляя новые сущности заданий в базу.

Целевая аудитория данного приложения – это студенты и все те, кто хочет обучаться. Цель приложения – обучить пользователя определенным темам, связанным с нечеткой логикой.

Приложение реализует ролевую систему пользования: пользователи, которые делятся на студентов и преподавателей, и администраторы. Пользователи-студенты получают доступ к личному кабинету и обучающему контенту, пользователи-преподаватели – к вышеперечисленному, а также к информации о пользователях-студентах. Администраторы получают права на редактирование информации в базе данных.

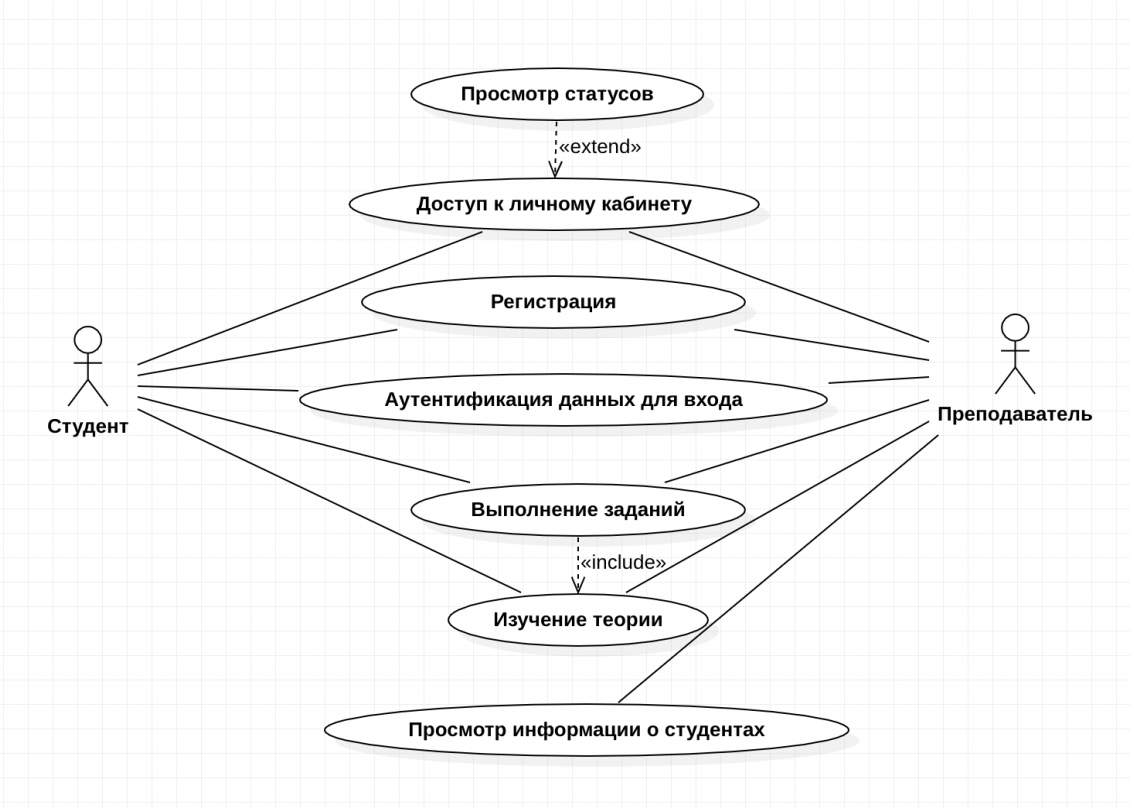


Рис. 1. Варианты использования приложения студентами и преподавателями

Приложение делится на две основные части – клиентскую и серверную. Клиентская часть отвечает за интерфейс приложения, к которому получают доступ все пользователи, серверная часть – за обработку запросов от клиентов и обновление информации в базе данных. Генерация заданий происходит на стороне сервера.

1.2. Средства реализации

Реализованное приложение является MERN-приложением [1-2], так как при разработке использовались следующие инструменты и средства реализации:

1. MongoDB [3-4] – документно-ориентированная СУБД. Создание документов велось в MongoDB Compass, клиенте для администрирования и просмотра данных.
2. Express [5] – фреймворк web-приложений для Node.js, спроектированный для создания веб-приложений и API.
3. React [6] – JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов в одностраничных и мобильных приложениях.
4. NodeJS [7] – программная платформа, расширяющая возможности языка JavaScript [8-9] и позволяющая использовать средства ввода-вывода через специальный API, подключать внешние библиотеки и т.д. В данном приложении на NodeJS была развернута серверная часть приложения.

1.3. Общее описание клиентской части приложения

Клиентская часть приложения реализована с помощью компонентного подхода фреймворка React. Все основные структурные единицы делятся на два типа – компоненты-элементы, готовые к переиспользованию в любой части приложения, и страницы-элементы, которые состоят из других компонентов.

Страницы нужны для перехода по основным частям приложения. Переходы осуществляются с помощью *роутинга*, механизма, при котором приложение меняет страницы в зависимости от текущего адреса в рамках приложения, на котором сейчас находится пользователь. Преподавателям и студентам доступен одинаковый набор страниц, контент которых меняется в зависимости от роли пользователя. Реализованы следующие страницы:

- личный кабинет пользователя: содержит основную информацию о пользователе и всех решенных/нерешенных заданиях (у преподавателя дополнительно имеется список студентов и их успеваемость);
- страница регистрации;
- страница авторизации;
- страница главной страницы: содержит список всех тем, доступных для изучения пользователем;
- страница с заданием (шаблон страницы является единым для всех заданий);
- страница с теоретической справкой (шаблон, аналогично странице с заданием, является единым для всех справок).

1.4. Общее описание серверной части приложения

Серверная часть включает в себя 3 основных элемента для обработки запросов и обновление информации в базе данных:

- API с набором конечных точек (endpoints);
- контроллеры – классы, устанавливающие взаимосвязь между конечными точками, адресами запросов и набором прав пользователей, для которых данные конечные точки доступны;
- база данных.

API включает в себя несколько конечных точек, разбитых на 2 класса контроллеров: authController (содержит точки, относящиеся к регистрации и аутентификации пользовательских данных) и studySystemController (содержит точки для получения и изменения заданий и теории). Система содержит следующие основные конечные точки:

- ‘/auth/registration’;
- ‘/auth/login’;
- ‘/auth/users’;
- ‘/auth/user-info’;
- ‘/study-center/task’ (GET-запрос);
- ‘/study-center/task’ (POST-запрос);
- ‘/study-center/tasks’;
- ‘/study-center/tasks-result’;
- ‘/study-center/theory’;
- ‘/study-center/theory-list’.

Аутентификация данных пользователей включает в себя JWT-токенизацию, механизм, предоставляющий пользовательскому клиенту секретный ключ-токен для возможности дальнейшего обмена информацией с сервером. Данный механизм был добавлен с целью обеспечения безопасности и предотвращения несанкционированного доступа в систему пользователей, не прошедших аутентификацию.

База данных является документно-ориентированной, поэтому является коллекцией, состоящей из набора документов. Схема базы данных представлена ниже на схеме (рисунок 2). Обратим внимание, что документ с заданиями отсутствует, так как задания являются объектами самой системы.

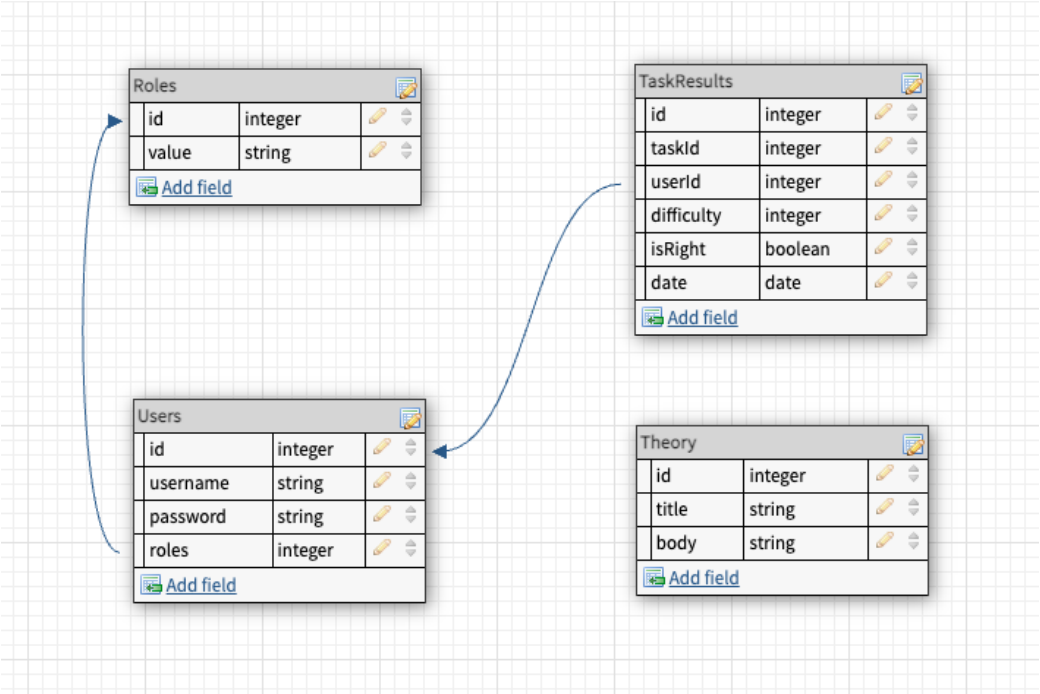


Рис. 2. Схема базы данных

2. Генерация тестовых заданий

Рассмотрим сценарий, при котором пользователь уже зарегистрировался в системе и находится на главной странице. Главная страница содержит все задания и теоретические справки, содержащиеся в системе и доступные определенным ролям пользователя.

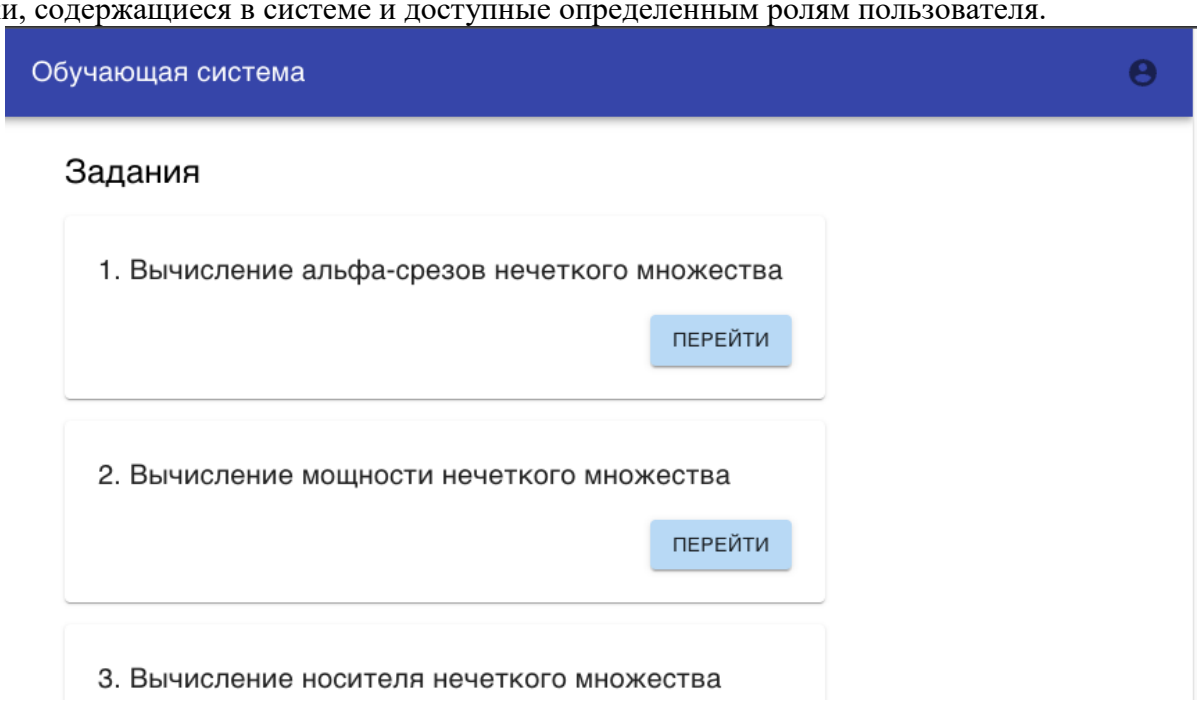


Рис. 3. Главное меню приложения

При переходе на любое задание посылается запрос к ‘/study-center/task’ с указанным id задания, после чего страница рендерит заголовок и тело задания с помощью инструмента математической верстки Latex.

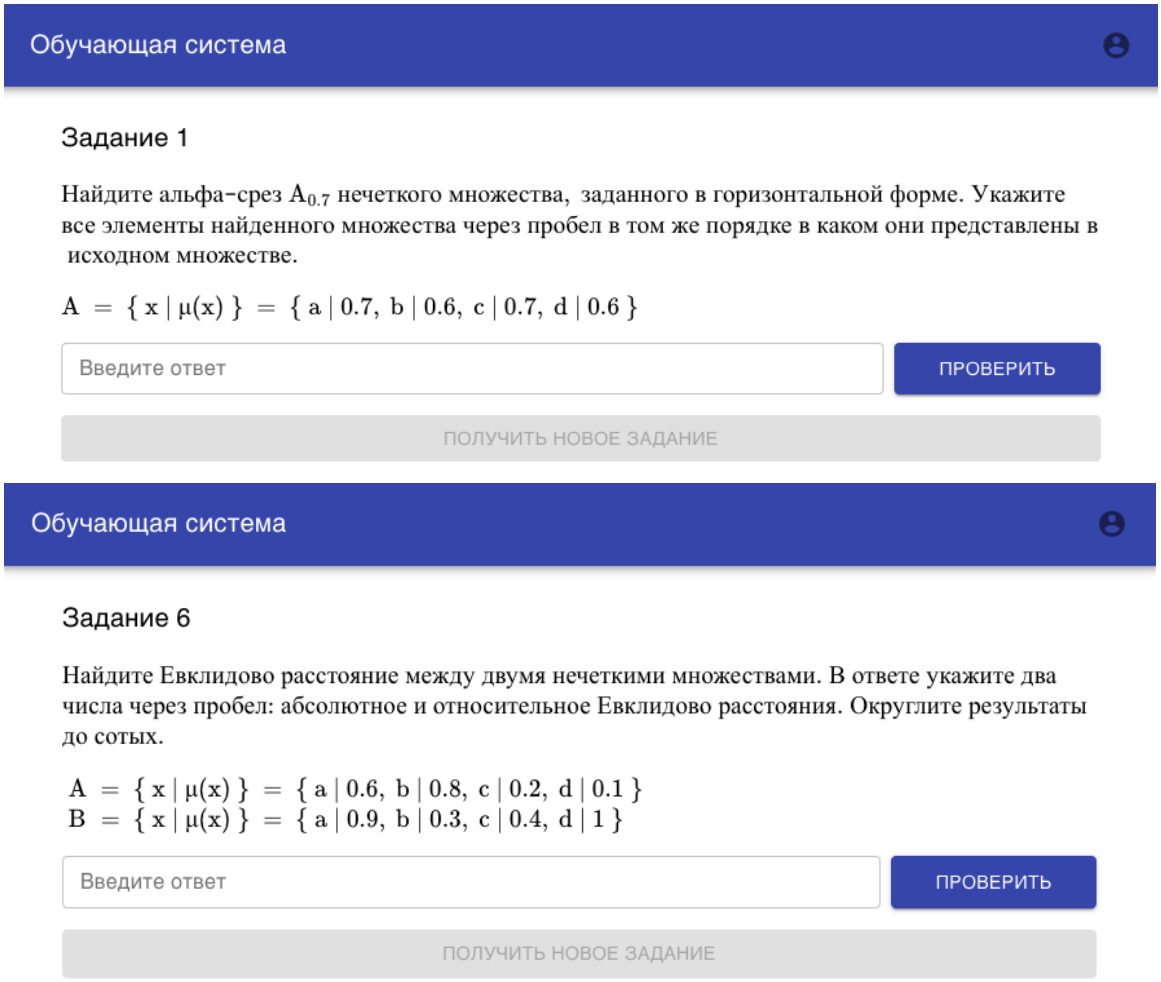


Рис. 4. Страницы со сгенерированными заданиями

Если пользователь нажмет кнопку «Проверить», то будет произведена проверка введенного ответа. Вне зависимости от того, был ли ответ правильным, система отправит запрос к ‘/study-center/task’ и информация будет сохранена в базу данных. Если задание было решено верно, то кнопка «Проверить» станет неактивной, а кнопка «Получить новое задание» станет активной. В противном случае – появится сообщение об ошибке, и пользователю будет доступна возможность решить задание еще раз.

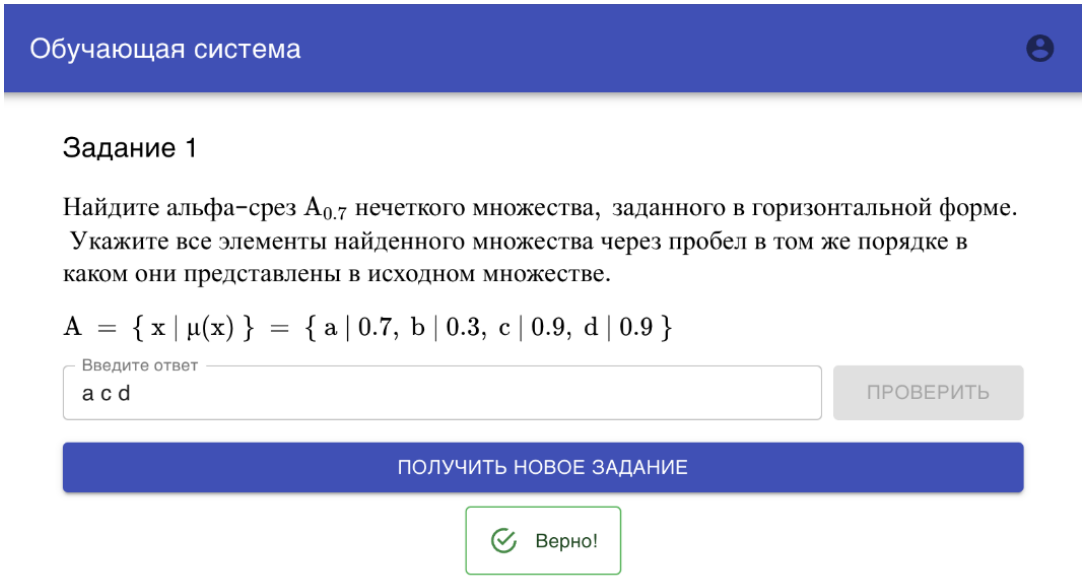


Рис. 5. Страница с верно решенным заданием

Заключение

В данной работе была описана реализация обучающего нечеткой логике клиент-серверного приложения с документно-ориентированной базой данных. Была реализована ролевая система пользования приложением, механизм JWT-токенов, а также API с набором конечных точек, на основе которых было создано два класса контроллеров.

Приложение позволяет реализовать дистанционный формат обучения. Также за счет автоматической генерации заданий по нечеткой логике часть ответственности за обучение студентов перекладывается на само приложение.

В перспективе планируется добавить новые задания по данной теме, вывод алгоритмов которых не является тривиальной задачей. Например, задания по методам оптимизации или задачи из области нечетких чисел.

Литература

1. Hoque, S. Full-Stack React Projects / S. Hoque. – Packt Publishing, 2020. – 716 p.
2. Lim, G. Beginning Node.js, Express & MongoDB Development / G. Lim. – 2019. – 135 p.
3. Электронный курс «MondoDB Basics» платформы MongoDB.Univercity. – Режим доступа: https://university.mongodb.com/courses/M001. – (Дата обращения: 13.01.2023).
4. Официальная документация API Mongoose. – Режим доступа: https://mongoosejs.com/docs/guide.html. – (Дата обращения: 13.01.2023).
5. Официальная документация Express. – Режим доступа: https://expressjs.com/en/resources/community.html – (Дата обращения: 12.01.2023).
6. Официальная документация React. – URL: https://react.dev/reference/react – (Дата обращения: 22.01.2023).
7. Официальная документация NodeJS. – URL: https://nodejs.org/docs/latest-v15.x/api/. – (Дата обращения: 12.01.2023).
8. Официальная документация JavaScript. – Режим доступа: http://www.ecma-international.org/ecma-262/10.0/index.html. – (Дата обращения: 22.01.2023).
9. Хавербеке, М. Выразительный JavaScript. Современное программирование / М. Хавербеке. – 3-е издание. – Санкт-Петербург, 2019. – 482 с.

УДК 004.4, 004.7

ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ SPRING FRAMEWORK ПРИ РАЗРАБОТКЕ СЕРВЕРНОЙ ЧАСТИ WEB-ПРИЛОЖЕНИЯ «РАБОЧЕЕ МЕСТО БИБЛИОТЕКАРЯ»

А. А. Ларина

Воронежский государственный университет

Введение

Современные web-приложения являются сложными системами, которые требуют масштабируемости, гибкости и высокой производительности. В связи с этим возникает потребность использования различных фреймворков, которые упрощают и ускоряют разработку, увеличивают стабильность приложения.

Одним из примеров таких фреймворков является Spring Framework. С его помощью можно создавать безопасные и масштабируемые приложения, а архитектурная гибкость при работе с данным фреймворком позволяет оптимизировать приложение с учетом конкретного задания.

В данной статье рассматриваются особенности использования Spring Framework при разработке серверной части web-приложения "Рабочее место библиотекаря".

1. Описание проекта «Рабочее место библиотекаря»

"Рабочее место библиотекаря" – это web-приложение, которое предназначено для автоматизации работы библиотекаря. Со стороны работника библиотеки оно позволяет хранить данные о книгах в удобном формате, управлять заявками на выдачу и возврат книг, журналов и другой литературы. Со стороны пользователя реализована возможность поиска интересующей литературы, онлайн подача заявки на книгу, составление списка прочитанного в личном профиле.

Приложение состоит из двух частей – клиентской и серверной. Клиентская часть представляет собой web-интерфейс для управления базой данных книг. Работник библиотеки может добавлять новые книги, редактировать сведения о текущих и осуществлять поиск и выдачу книг. Серверная часть выполняет следующие функции:

1. Управление базой данных: серверная часть обрабатывает запросы клиентской части и осуществляет доступ к базе данных, где хранится информация о книгах, читателях, выданных книгах и других элементах библиотечного фонда.
2. Авторизация и аутентификация: серверная часть обеспечивает безопасность данных и контролирует доступ к системе. Она проверяет логин и пароль пользователя, а также устанавливает права доступа к различным функциям приложения.
3. Обработка запросов пользователей: серверная часть обрабатывает запросы, поступающие от клиентской части, и возвращает результаты выполнения операций. Например, при запросе на поиск книги по автору или жанру серверная часть выполняет поиск в базе

данных и возвращает список соответствующих книг.

4. Синхронизация данных: серверная часть обеспечивает синхронизацию данных между клиентскими приложениями и базой данных. Это позволяет пользователям работать с актуальной информацией и избежать конфликтов при одновременном доступе к данным.

5. Обновление и поддержка приложения: серверная часть обеспечивает обновление и поддержку приложения, включая исправление ошибок, добавление новых функций и техническую поддержку.

2. Этапы разработки

Разработка web-приложения была разделена на несколько этапов:

1. Анализ требований: определение функциональных и нефункциональных требований к приложению, а также определение потенциальных пользователей и их потребностей.
2. Проектирование приложения: создание дизайна интерфейса, выбор технологий и инструментов, разработка архитектуры приложения.
3. Разработка клиентской части: создание пользовательского интерфейса, реализация методов для взаимодействия с сервером.
4. Разработка серверной части: создание базы данных, реализация методов для обработки запросов клиентской части, обеспечение безопасности данных.
5. Тестирование: проверка работоспособности и стабильности приложения, выявление и исправление ошибок.
6. Публикация проекта.

3. Анализ Spring Framework

3.1. Краткое описание

Серверная часть была реализована на языке Java 17 версии, с использованием фреймворка Spring Boot. Он предоставляет набор инструментов и библиотек для упрощения процесса создания и развертывания приложений. При разработке были использованы следующие технологии:

1. Spring Security. Модуль для обеспечения безопасности приложения, который предоставляет механизмы аутентификации и авторизации, что позволяет управлять доступом пользователей к различным частям приложения.

2. Spring Data JPA. Модуль, который предоставляет удобный и простой способ работы с базой данных. Он позволил сократить количество кода, необходимого для работы с базой данных, благодаря автоматической генерации SQL-запросов.

3. Spring MVC. Модуль, при помощи которого обрабатываются rest-запросы, возвращая результат в виде HTML-страницы.

3.2. Инструментарий

Сервер Apache Tomcat, встроенный в фреймворк, был использован в качестве web-сервера. Для хранения данных была использована база данных MySQL. Данная СУБД была выбрана, поскольку MySQL легко интегрируется с Spring Framework, а также обладает мощными механизмами безопасности, что защищает данные от несанкционированного доступа и взлома. Надежная защита данных стала одной из ключевых причин выбора СУБД MySQL, поскольку после регистрации пароли пользователей хранятся в базе данных. Также необходимо учитывать, что разработка данного веб-приложения не предполагает хранение огромного количества данных, а следовательно использование платных, промышленных СУБД избыточно. MySQL является одной из наиболее производительных СУБД для небольших данных, а её кроссплатформенность исключает привязку к определенному типу устройств. Важно также отметить, что к данной СУБД уже существует драйвер, соответствующий стандарту JDBC, позволяющий наладить общение между приложением на Spring Framework и используемой базой данных без особых трудностей, а поддержка стандарта JPA позволяет работать со строками из таблицы как с Java-объектами.

Для работы с данными был выбран ORM Framework Hibernate. При помощи него было реализовано сохранение, извлечение и обновление объектов в базе данных с помощью операций CRUD (Create, Read, Update, Delete). Основными достоинствами данного фреймворка являются:

1. Упрощение разработки приложения. Hibernate позволяет работать с базой данных на уровне объектов, что позволяет избежать написания SQL запросов.
2. Повышение производительности приложения.
3. Безопасность. Hibernate предоставляет механизмы для защиты приложения от атак типа SQL injection.

4. Реализация проекта

4.1. Проблематика

В результате сравнительного анализа с аналогичными приложениями данной предметной области были обнаружены два основных их недостатка. В первую очередь, отсутствие функций для пользователя — читателя библиотеки. Например, пользователь не может подать заявку на интересующую книгу в режиме онлайн. Кроме того, все реализованные решения поддерживаются только на английском языке, что значительно снижает возможность их использования в отечественных библиотеках. Таким образом, реализация интерфейса приложения на русском и расширение возможностей пользователя позволяют оптимизировать работу библиотек и привлечь новых посетителей.

4.2. Разработка архитектуры приложения

Перед началом реализации проекта была разработана архитектура приложения, которая определяла структуру проекта и его компоненты. Архитектура приложения включала в себя следующие компоненты:

1. Контроллеры (Controllers): отвечают за обработку HTTP-запросов, получаемых

от клиента, и управление процессом обработки запросов.
2. Сервисы (Services): содержат бизнес-логику приложения и обеспечивают взаимодействие между контроллерами и репозиториями.
3. Репозитории (Repositories): отвечают за взаимодействие с базой данных и предоставляют методы для получения и сохранения данных.
4. Модели (Models): представляют собой объекты данных, которые используются в приложении для хранения информации.

4.3. Разработка базы данных

Разработанная схема базы данных включает в себя следующие таблицы:

1. Книги (Books): содержит информацию о книгах, доступных в библиотеке.
2. Пользователи (Users): содержит информацию о пользователях (логин, пароль и т.д.).
3. Заявка (Book application): содержит информацию о заявке на книгу (название книги, id пользователя, дату).
4. Роли (Roles): содержит информацию о роли пользователя (администратор и пользователь).

Заключение

В результате анализа средств back-end разработки веб-приложений были выбраны инструменты, необходимые для реализации серверной части проекта. Разработана структура и реализованы функциональные возможности web-приложения "Рабочее место библиотекаря".

Литература

1. Документация Spring Boot. – Режим доступа: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ (Дата обращения: 28.03.2023)
2. Документация Hibernate ORM. – Режим доступа: https://hibernate.org/orm/documentation/6.1/ (Дата обращения: 28.03.2023)
3. Spring Boot 101: Введение в создание веб-приложений. – Режим доступа: https://vc.ru/u/1389654-machine-learning/586955-spring-boot-101-vvedenie-v-sozdanie-veb-prilozheniy (Дата обращения: 28.03.2023)

УДК *004*

УМНОЕ УСТРОЙСТВО ДЛЯ РАЗЛИТИЯ НАПИТКОВ «НАЛИВАТОР»

А. Н. Леденев

Воронежский государственный университет

Введение

В настоящее время сложно найти современную квартиру или дом без автоматической системы. Нас все чаще и чаще окружают умные розетки, умные светильники, робот-пылесосы и многие другие гаджеты. Они часто облегчают нашу жизнь.

В связи с такой популярностью умных гаджетов человек стремится автоматизировать чуть ли ни каждый процесс. Одним из таких устройств является «Наливатор». Это настольный прибор, которых украсит любой праздник и сделает его легче.

Наливатор представляет собой технически совершенное устройство, предназначенное для разлива любых жидких напитков в стаканчики с высокой точностью (до миллилитра). Устройство обладает настройками стороны налива, которые позволяют определить направление разлива, а также автоматическим и ручным режимами работы. В дополнение к этому, «Наливатор» оснащен счетчиком общего и сбрасываемого «пробега», что позволяет контролировать количество разлитых напитков.

Управление устройством осуществляется с помощью энкодера, расположенного на корпусе, а также через Bluetooth с телефона на операционной системе Android. Интерфейс управления прост и понятен, что делает использование «Наливатора» удобным и доступным для любого пользователя. Прибор обладает не только функциональностью, но и имеет современный дизайн. Он также занимает минимум места и обеспечивает безопасность использования. Это устройство, которое точно пригодится в любой домашней коллекции умных гаджетов.

Управляется устройство с помощью микроконтроллера ATmega328 на плате Arduino. Использование данной платы дает множество преимуществ, таких как гибкость и возможность настройки под различные задачи, широкий выбор дополнительных модулей и сенсоров, наличие библиотек и примеров кода для быстрого создания проектов, низкое энергопотребление.

1. Проектирование и сборка аппаратной части

Одной из главных задач при разработке «Наливатора» было определение количества стаканов, которые он может разливать. В результате, было решено, что устройство может работать с 1 до 5 стаканами одновременно. Кроме того, важным моментом является определение наличия стакана. Для этого была предусмотрена специальная система датчиков, которая позволяет определить наличие стакана и автоматически начать процесс разлива напитка.

Другой важный вопрос – это материал корпуса. Было принято решение изготовить корпус из фанеры, покрытую бесцветным лаком, которая не только прочная, но и безопасная для использования.

В целях объединения всей электроники в одном месте была создана печатная плата в рамках программного обеспечения Sprint Layout 7, после чего она была заказана на заводе

JLCPCB. В итоге была получена двухслойная печатная плата высокого качества, которая полностью готова для пайки электронных компонентов (рис. 1).

Поскольку водяной насос работает при напряжении 12 вольт, а управляющая часть функционирует на напряжении 5 вольт, в схему был добавлен стабилизатор напряжения 12-5 вольт. Для обеспечения оптимального размещения стаканов в корпусе были отфрезерованы соответствующие установочные места. В дальнейшем в эти места были добавлены датчик препятствия и RGB светодиод. Каждый светодиод имеет 4 вывода и для оптимизации управления ими были использованы сдвиговые регистры 74HC595.

Выходной кран, к которому подведен шланг, закреплен на сервоприводе, что позволяет осуществлять налив на 180 градусов

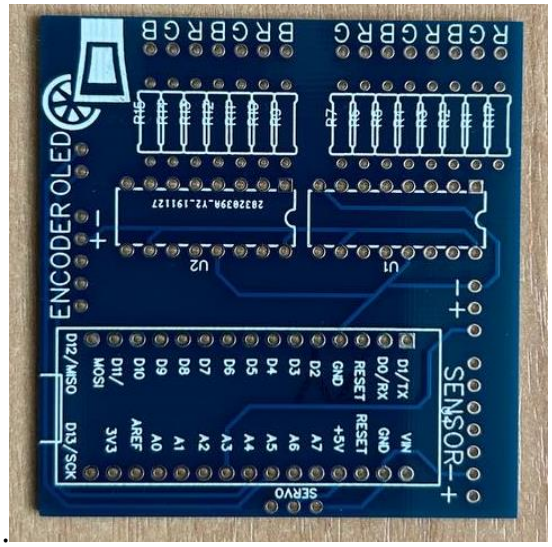


Рис. 1. Печатная плата

2. Программирование микроконтроллера и разработка Android приложения

Arduino имеет простой и понятный язык программирования, основанный на C++, который позволяет разработчикам быстро и легко создавать программы для управления различными устройствами. Для программирования Arduino используется среда разработки Arduino IDE, которая предоставляет все необходимые инструменты для создания и отладки программ.

Основной задачей в написании программы для данного устройства была разработка интуитивно понятного интерфейса для пользователя, а также удобного управления устройством с помощью энкодера. Для этого используются различные библиотеки, которые регистрируют различные сигналы и энкодера, такие как: нажатие, удерживание, поворот влево, поворот вправо, поворот влево с нажатием, поворот вправо с нажатием. Все эти команды позволили организовать управление всем устройством.

Также немаловажно было написать функцию для обнаружения стаканов с советующей индикацией: светодиод не горит – стакан отсутствует, синий цвет – стакан обнаружен, красный цвет – происходит процесс налива, зеленый цвет – налив завершен.

В программе используется библиотека EPPROM для работы с энергонезависимой памятью Arduino. В ней хранятся настройки стороны, режима и количество налитой жидкости за весь период использования.

Как дополнительное устройство для управления «Наливатором» может использоваться мобильный телефон с операционной системой Android. Для этого было разработано несложное приложение, но полностью готовое для управления всеми функциями устройства.

В качестве среды разработки и языка программирования был выбрал Delphi. К преимуществам данной среды разработки относят:

- Поддержка множества операционных систем, включая iOS, Android и Windows;
- Быстрый и эффективный компилятор;
- Интеграция с другими инструментами разработки Embarcadero, такими как RAD Studio и C++ Builder;
- Широкий выбор компонентов и библиотек для упрощения разработки;

Для передачи данных между телефоном и Arduino используется Bluetooth. В качестве команд был разработан собственный протокол связи. Каждая команда имеет структуру <ключ><значение>. Например, команда для запуска процесса налива выглядит как P50, где 50 – это количество миллилитров, которые нужно налить. Каждая команда отправляется в формате пакета байтов на устройство Arduino. Далее, платформа конвертирует эти данные в строковый тип и выполняет соответствующее задание, которое было указано пользователем.

Заключение

В результате работы получаем полностью работоспособное устройство, которое отвечает требованиям потребителя. Тестирование показало, что ошибок в ходе работы минимальное количество и устройство полностью готово к эксплуатации. (рис. 2).

Рис. 2. «Наливатор»

«Наливатор» может использоваться как помощник для людей с ограниченными возможностями, он способен налить жидкость в стакан без особых усилий человека.

Кроме того, данное устройство может быть использовано в кафе, ресторанах и других заведениях общепита для автоматизации процесса налива напитков. В целом, «Наливатор» является эффективным и инновационным решением для автоматизации процесса налива напитков в различных сферах бизнеса. Благодаря своей компактности и простоте использования, «Наливатор» может стать популярным и востребованным устройством на рынке.

«Наливатор» – это устройство, которое точно пригодится в любой домашней коллекции умных гаджетов. Он позволяет экономить время и силы при разливе напитков на мероприятиях и создает атмосферу комфорта и уюта. Благодаря своему дизайну и компактности, «Наливатор» легко интегрируется в любой интерьер и может стать отличным подарком для любого человека, который ценит удобство и функциональность (рис. 2).

Литература

1. Sprint Layout 5. Подробное руководство. – Режим доступа: https://easyelectronics.ru/sprint-layout-5-podrobnoe-rukovodstvo.html. (Дата обращения: 08.09.2022).
2. Программирование Ардуино. – Режим доступа: https://arduino.ru/Reference. – (Дата обращения: 25.11.2022).
3. Осипов, Д. Л. Delphi. Программирование для Windows, OS X, iOS и Android. / Осипов Д. Л. — СПб.: БХВ-Петербург, 2014. — 464 с.
4. Использование Bluetooth в Delphi. – Режим доступа: https://docwiki.embarcadero.com/RADStudio/Alexandria/en/Using_Classic_Bluetooth. (Дата обращения: 12.12.2022).
5. Программирование на Delphi. Разработка Android-приложения. – Режим доступа: https://progtips.ru/delphi/programmirovanie-na-delphi-chast-2-razrabotka-android-prilozheniya.html. (Дата обращения: 11.12.2022)

УДК 517.97: 532.526

РЕАЛИЗАЦИЯ СТРАТЕГИИ УПРАВЛЕНИЯ ВЫВОДОМ НА ОСНОВЕ МИНИМАЛЬНОГО ДИЗЪЮНКТА

М. В. Лещинская

Воронежский государственный университет

Введение

Интеллектуальные системы используют разные модели представления знаний. Логическая модель на основе исчисления предикатов первого порядка обладает объяснительной способностью и строгим теоретическим обоснованием, что полезно при разработке систем с объяснениями. Если предметная область содержит жесткие знания, то использование логической модели и алгоритмов логического вывода целесообразно. Метод резолюций является одним из основных методов исчисления предикатов первого порядка, но полный перебор, используемый для построения логического вывода, является ресурсозатратным в реальных задачах [1, 2]. Для повышения эффективности метода резолюций были разработаны различные стратегии управления выводом [3, 4].

В данной статье представлена реализация стратегии управления выводом на основе минимального дизъюнкта [5] на языке C++. Результаты исследования помогут убедиться в повышении эффективности поиска решения с помощью данного алгоритма и начать его использовать.

1. Стратегия управления выводом на основе минимального дизъюнкта

Теоретическая база исследований описана в [5] достаточно подробно ниже приведена выдержка алгоритма из [5], которая позволяет сравнить задуманное с приведенной далее реализацией.

Пусть S – исходное множество дизъюнктов; R – подмножество дизъюнктов из S, которое обрабатывается на определенном этапе алгоритма; S'_k, S''_k – два множества предложений, которые построены из S_k на k-ой итерации; d – дизъюнкт из S'_k; ad – дизъюнкт из S''_k, который выбирается для построения резольвенты (этот дизъюнкт будем называть активным).

Приведем описание алгоритма.

Шаг 1 (инициализация). Положить $k = 0$, $D = S$.

Шаг 2 (правило 1 формирования множеств S'_k и S''_k). Выбрать любой минимальной дизъюнкт из множества D и поместить его в множество S'_k. Положить $S''_k = D \setminus S'_k$.

Шаг 3 (критерии останова). Если S''_k существуют дизъюнкты, резольвента которых образует пустой дизъюнкт □, то останов (исходное множество S является противоречивым). Иначе проверить: если $D = \varnothing$, то останов (исходное множество S является выполнимым); если $D \neq \varnothing$, то перейти к следующему шагу.

Шаг 4 (построение резольвенты). Если существует резольвента $res(d, ad)$ дизъюнктов $d \in S'_k$ и $ad \in S''_k \setminus d$ в форме минимального дизъюнкта или такая, которая является единственно

возможной, то перейти к следующему шагу. Иначе положить $D = S_k''$, $k = k+1$ и перейти к шагу 2.

Шаг 5 (правило 2 формирования множеств S_k' и S_k''). Положить $k = k+1$, $S_k' = \{res(d, ad)\}$, $S_k'' = (S_k' \setminus ad) \cup \{d\}$ и перейти на шаг 3.

2. Реализации стратегии на основе минимального дизъюнкта

Опираясь на описание стратегии поиска минимального дизъюнкта и схему реализации алгоритма (рис 2.1), продумана программная реализация. Для имплементации алгоритма необходимо иметь такие объекты и функции как атом и дизъюнкт, функция поиска минимального дизъюнкта, а также функции для работы с резольвентой: поиск дизъюнктов, которые могут построить резольвенту и построение самой резольвенты.

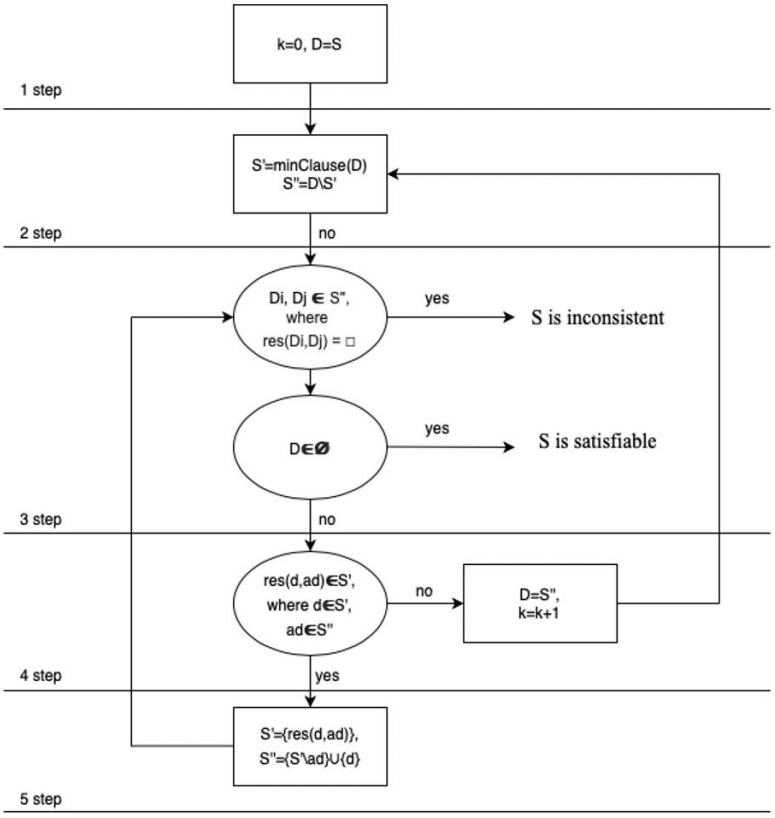


Рис. 2.1. Схема работы алгоритма

В листинге 2.1 представлены структуры для атома и дизъюнкта: каждый атом обладает или не обладает отрицанием, а также имеет наименование, каждый дизъюнкт состоит из нескольких атомов. Соответственно, программа принимает данные в вектор дизъюнктов, который состоит из некоторого количества атомов.

В листинге 2.2 описана функция, которая возвращает минимальный дизъюнкт в наборе m, который является вектором дизъюнктов.

Листинг 2.1. Структуры атома и дизъюнкта

```
struct Atom {
   int otr;
   string s;
};

struct Clause {
   int si;
   vector<Atom> a;
```

Листинг 2.2. Функция, возвращающая мин.дизъюнкт в наборе

```
Clause findMinD(vector<Clause> &m, int n){
   Clause minClause=m[0]; int minSi=minClause.si;
   for (int i=0; i<n; i++) {
      if (m[i].si<minSi) {
         minSi=m[i].si;
         minClause=m[i];
      }
   }
```

В листингах 2.3-2.5 описаны функции для работы с резольвентой. Функция 2.3 bool notEquals(Clause a, Clause b) возвращает bool-значение true, если два дизъюнкта не равны. Равенство происходит через сравнение каждого атома и его отрицания. Функция 2.4 bool resolventExists(Clause d, Clause ad) возвращает bool-значение true, если резольвента может быть построена между дизъюнктами d и ad. Функция 2.5 Clause resolvent(Clause d, Clause ad) возвращает резольвенту, которую образовали два дизъюнкта d и ad. В резолютирующий дизъюнкт попадают все атомы, которые не являются атомом X и атомом не X.

Листинг 2.3. Функция, которая возвращает true, если два дизъюнкта отличаются

```
bool notEquals(Clause a, Clause b) {
   if (a.si!=b.si) return true;
   for (int i=0; i<a.si; i++) {
      if (a.a[i].s!=b.a[i].s ||
a.a[i].otr!=b.a[i].otr) return true;
   }
   return false;
};
```

Листинг 2.4. Функция, которая возвращает true, если два дизъюнкта могут построить резольвенту

```
bool resolventExists(Clause d, Clause ad) {
   for (int i=0; i<d.si; i++) {
      for (int j=0; j<ad.si; j++) {
         if (d.a[i].s==ad.a[j].s &&
d.a[i].otr!=ad.a[j].otr) return true;
      }
   }
   return false;
};
```

Функция 2.6 int minimalClause(vector<Clause> vD, int n) является основной для работы

Листинг 2.5. Функция, строит резольвенту и возвращает ее

```
Clause resolvent(Clause d, Clause ad) {
   Clause res; res.si=0;
   for (int i=0; i<ad.si; i++) {
      for (int j=0; j<d.si; j++) {
         if (d.a[j].s==ad.a[i].s && d.a[j].otr!=ad.a[i].otr) {
            for (int i2=0; i2<ad.si; i2++) {
               if (i2!=i) {
                  Atom a; a.s=ad.a[i2].s; a.otr=ad.a[i2].otr;
                  res.a.push_back(a); res.si++;
               }
            }
            for (int j2=0; j2<d.si; j2++) {
               if (j2!=j) {
                  Atom a; a.s=d.a[j2].s; a.otr=d.a[j2].otr;
                  res.a.push_back(a); res.si++;
               }
            }
         }}}
   return res;}
```

алгоритма, основанного на поиске минимального дизъюнкта.

Последовательно имплементируются шаги алгоритма: выполняем цикл до тех пор, пока, не найдем пустой дизъюнкт (а значит множество противоречиво) или множество, в котором ищем минимальный дизъюнкт, не станет пустым (а значит множество выполнимо).

Листинг 2.6. Основная функция, реализующая стратегию поиска минимального дизъюнкта

```
int minimalClause(vector<Clause> vD, int n) {
   int count=0;
   vector<Clause> s1,s2;
   bool fVD = false;
   while (count<=LIMIT) {
      count++;
      s1.push_back(findMinD(vD, n));
      for (int i=0; i< vD.size(); i++) {
         for (int i1=0; i1<s1.size(); i1++) {
            if (notEquals(vD[i], s1[i1]))s2.push_back(vD[i]);
         }}
          cout<<"\n--RESOLVENT--";
      for (int i1=0; i1<s1.size() && count<LIMIT; i1++) {
         for (int i2=0; i2<s2.size(); i2++) {
            if (resolventExists(s1[i1], s2[i2])) {
               Clause res = resolvent(s1[i1], s2[i2]);
               if (res.si==0) {
                  cout<<"¤";
                  cout<<"\n\n ANSWER = INCONSISTENT\n";
                  return 0;
               }
               printClause(res);
               s1.push_back(res);
               s2.erase(s2.begin()+i2);s2.push_back(s1[i1]);
               fVD=true;
            }
            count++;
         }
      }
      if (vD.size()==0) {
         cout<<"\n\n ANSWER = SATISFIABLE\n";
         return 0;
      }

      if (!fVD) {
         n=s2.size(); s2.swap(vD);
         vector<Clause>().swap(s2); vector<Clause>().swap(s1);
      }
   }
   cout<<"\n\n ANSWER = MAYBE SATISFIABLE\n";
   return 0;
}
```

Внутри функции собираем множество S1 и множество S2, которые в программе являются векторами дизъюнктов. Перебираем значения из векторов, если d из вектора S1 и ad из вектора S2 могут построить резольвенту (**if** (resolventExists(s1[i1], s2[i2])), то она строится (Clause res = resolvent(s1[i1], s2[i2])), если был получен пустой дизъюнкт, то программа завершает свою работу, получаем ответ (cout<<"\n\n ANSWER = INCONSISTENT\n";).

Иначе изменяем вектора S1 и S2.

Проверяем основной вектор на пустоту, если он пустой, то получаем ответ (cout<<"\n\n ANSWER = SATISFIABLE\n"), иначе меняем его.

Переменная LIMIT указана для предотвращения проблемы зацикливания, ограничение по-умолчанию стоит на 100 итераций.

3. Вычислительные эксперименты

Пример 1. Пусть задано следующее множество предложений:

$$S=\left\{C(x)\vee O(x),B(y)\vee\bar{C}(y),W(A),\bar{B}(A),\bar{O}(z)\vee\bar{W}(z),C(v)\vee B(v)\vee\bar{W}(v)\right\}.$$

Данные для ввода:

6
2 0 C 0 O
2 0 B 1 C
1 0 W
1 1 B
2 1 O 1 W
3 0 C 0 B 1 W

На рис. 3.1 (слева) представлен пример работы алгоритма, основанного на поиске минимального дизъюнкта. Видим, что множество S – противоречиво. Так же (справа) отображено время работы алгоритма поиска минимального дизъюнкта и полного перебора. Видно, что новая стратегия работает эффективнее по времени.

```
6
2 0 C 0 O
2 0 B 1 C
1 0 W
1 1 B
2 1 O 1 W
3 0 C 0 B 1 W
C∨O, B∨¬C, W, ¬B, ¬O∨¬W, C∨B∨¬W,
--RESOLVENT--
res(a,ad)=res(W, ¬O∨¬W) = ¬O
res(a,ad)=res(¬O, C∨O) = C
res(a,ad)=res(C, B∨¬C) = B
res(a,ad)=res(B, ¬B) = □

 ANSWER = INCONSISTENT
```

```
6
2 0 C 0 O
2 0 B 1 C
1 0 W
1 1 B
2 1 O 1 W
3 0 C 0 B 1 W
C∨O, B∨¬C, W, ¬B, ¬O∨¬W, C∨B∨¬W,

 ANSWER = INCONSISTENT

MinimalClause TIME = 0.063
ExhaustiveSearch TIME = 0.127
```

Рис. 3.1. Пример работы алгоритма на Примере 1

Пример 2. Теперь рассмотрим следующее множество

$$S=\left\{C(x)\vee O(x)\vee\bar{B}(x),B(y)\vee C(y),W(A)\vee\bar{B}(A),\bar{O}(z)\vee\bar{W}(z),\bar{C}(v)\vee B(v)\vee\bar{W}(v)\right\}.$$

По сравнению с предыдущим примером множество S не содержит одночленов. Данное множество выполнимо, и □ получить нельзя.

Данные для ввода:
5
3 0 C 0 O 1 B
2 0 B 0 C
2 0 W 1 B
2 1 O 1 W
3 1 C 0 B 1 W

На рис. 3.2 представлен пример работы алгоритма, основанного на поиске минимального дизъюнкта. Видим, что множество S – выполнимо.

```
5
3 0 C 0 O 1 B
2 0 B 0 C
2 0 W 1 B
2 1 O 1 W
3 1 C 0 B 1 W

C∨O∨¬B, B∨C, W∨¬B, ¬O∨¬W, ¬C∨B∨¬W,
--RESOLVENT--
res(a,ad)=res(B∨C, C∨O∨¬B) = C∨O∨C
NOres(a,ad)=res(B∨C, W∨¬B)
NOres(a,ad)=res(B∨C, ¬O∨¬W)
res(a,ad)=res(B∨C, ¬C∨B∨¬W) = B∨¬W∨B
NOres(a,ad)=res(B∨C, B∨C)
NOres(a,ad)=res(B∨C, B∨C)
NOres(a,ad)=res(C∨O∨C, W∨¬B)
res(a,ad)=res(C∨O∨C, ¬O∨¬W) = ¬W∨C∨C
NOres(a,ad)=res(C∨O∨C, B∨C)
NOres(a,ad)=res(C∨O∨C, B∨C)
NOres(a,ad)=res(C∨O∨C, C∨O∨C)
res(a,ad)=res(B∨¬W∨B, W∨¬B) = ¬B∨B∨B∨W∨¬W∨B∨W∨B∨¬W
NOres(a,ad)=res(B∨¬W∨B, B∨C)
NOres(a,ad)=res(B∨¬W∨B, B∨C)
NOres(a,ad)=res(B∨¬W∨B, C∨O∨C)
NOres(a,ad)=res(B∨¬W∨B, B∨¬W∨B)

 ANSWER = MAYBE SATISFIABLE
```

Рис. 3.2. Пример работы алгоритма на Примере 2

Пример 3. Рассмотрим следующее множество

$$S = \left\{ C(x) \vee O(x), B(y) \vee \overline{C}(y), W(A), \overline{B}(A), \overline{O}(z) \right\}.$$

Данные для ввода:
5
2 0 C 0 O
2 0 B 1 C
1 0 W
1 1 B
1 1 O

В данном примере по сравнению с предыдущими уже на начальных итерациях потребуется изменение множества D. На рис. 3.3 представлен пример работы алгоритма, основанного на поиске минимального дизъюнкта. Видим, что множество S – противоречиво.

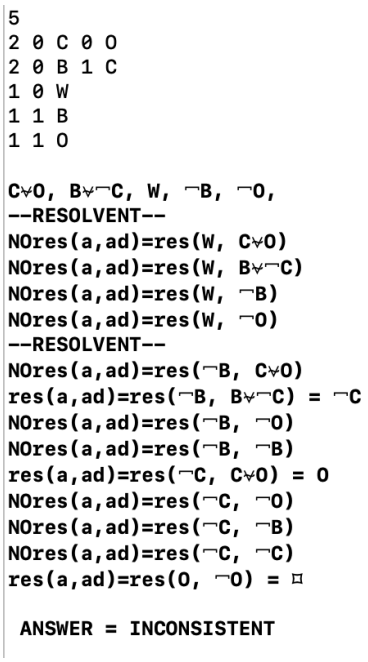


Рис. 3.3. Пример работы алгоритма на Примере 3

Таким образом, практически показана работа стратегии, алгоритм, который основан на поиске и работе с минимальным дизъюнктом и получены корректные результаты.

Заключение

Данная статья описывает эффективную стратегию на основе поиска минимального дизъюнкта, которая значительно улучшает производительность по сравнению со стратегией полного перебора. Реализация алгоритма поиска минимального дизъюнкта показала высокое качество и скорость работы на ряде вычислительных экспериментов.

Литература

1. Маслов С. Ю. Обратный метод установления выводимости в классическом исчислении предикатов / Маслов С. Ю. // ДАН СССР, 1964. – Вып. 159. – № 1. – С. 17-20.

2. Леденева, Т. М. Формальные аксиоматические теории. Исчисление предикатов. Часть 1. / Т.М. Леденева, Е.М. Аристова. – Воронеж: Издательский дом ВГУ, 2016. – 34 с.

3. Леденева Т. М. Формальные аксиоматические теории. Исчисление предикатов. Часть 2. / Т.М. Леденева. – Воронеж: Издательский дом ВГУ, 2020. – 45 с.

4. Леденева Т.М. Метод резолюций и стратегии поиска опровержений / Т.М. Леденева, М.В. Лещинская // Вестник ВГУ. Серия: Системный анализ и информационные технологии, 2021. – № 1. – С. 98-111.

5. Леденева Т.М. Новая стратегия поиска опровержений метода резолюций и алгоритмы её реализации / Т.М. Леденева, М.В. Лещинская // Актуальные проблемы прикладной математики, информатики и механики. Воронеж, Издательство «Научно-исследовательские публикации», 2022. – С. 1628-1633.

УДК: 004.4'22

ПРОБЛЕМЫ АВТОМАТИЗАЦИИ РОССИЙСКИХ ПРЕДПРИЯТИЙ НА ПЛАТФОРМЕ 1С:ПРЕДПРИЯТИЕ 8

А.Л. Майзель

Воронежский государственный университет

Введение

Деятельность современного предприятия невозможно представить без использования компьютерных систем, автоматизирующих различные аспекты его деятельности. Целью настоящей статья является анализ аспектов влияющих на внедрение проектов класса ERP на российских предприятиях, автоматизацию их финансовых и производственных контуров, анализ «критических» точек проекта.

Согласно проводившимся в предыдущие годы исследованиям рынка бизнес-систем, решения от компании «1С» прочно удерживали первое место по количеству внедрений, в настоящее время отрыв лишь увеличился. Необходимо отметить, технологичность, функциональный охват, производительность и способность к масштабированию. Именно по этой причине в данной статье предлагается сосредоточиться на особенностях внедрения применительно к продуктам компании «1С».

1. Постановка задачи

В качестве типового сценария можно рассматривать потребность предприятия в переходе с устаревшей или же не поддерживаемой вендором информационной системы на современную. Инициация подобного перехода подразумевает в первую очередь разработку дорожной карты будущих мероприятий. При этом ощутимыми препятствиями в реализации проекта могут оказаться:

- отсутствие у Заказчика глубоких знаний о внедряемой информационной системе, методологии работы с ней,
- попытка сотрудников Заказчика изменить поведение будущей системы под свой пользовательский опыт, полученный в рамках работы со старой системой,
- низкое качество нормативно-справочной информации исходной информационной системы,
- отсутствие проработанных бизнес-процессов,
- накопленные артефакты данных в исходной базе,
- размер проекта и масштабность внедрения устаревшей информационной системы,
- неверная этапность реализации проекта,
- бюджет проект и сроков реализации и тд.

2. Этапы проекта

Была выработана определенная дорожная карта работы над проектом автоматизации, позволяющая минимизировать риски неуспеха, разумно распределить силы проектной команды, соблюсти сроки и бюджет проекта. Опишем этапы верхнего уровня. Все этапы могут выполняться в несколько итераций, последовательно уточняя ожидаемый результат.

2.1 Формализация требований к бизнес-процессам

Первый этап работы над проектом автоматизации предполагает описание всех существенных бизнес-процессов предприятия, подлежащих реализации в новой информационной системе. Крайне важно, чтобы в описании участвовал компетентный представитель предприятия, имеющий полномочия изменения сложившихся практик при реинжиниринге существующих процессов, а со стороны внедренца – аналитик с глубоким знанием возможностей внедряемого программного продукта. Результатом описания станет согласованный набор формализованных требований к функциональности будущей системы. Важно, что эта формализация происходит на языке понятном бизнес-пользователям.

2.2 Моделирование

Программные продукты «1С», такие как «ERP», «Комплексная автоматизация», «Управление холдингом» и тд. – системы, состоящие из значительного количества взаимосвязанных между собой контуров. Этап Моделирования предполагает воссоздание всех бизнес-процессов предприятия, описанных на предыдущем этапе в выбранном программном продукте. При этом хорошей практикой являлась бы, возможно, частичная или полная загрузка исторических данных из существующей информационной системы. Моделирование позволяет преодолеть барьер непонимания аспектов работы новой информационной системы в глазах пользователя, провести GAP-анализ и зафиксировать потребности в изменении или привнесении новых функциональных возможностей в систему.

Перенос данных – позволяет по-новому взглянуть на качество и актуальность существующей НСИ. На этом этапе имеет смысл определиться с последовательность внедрения тех или иных подсистем будущего решения. Зачастую невозможно одномоментно перестроить работу предприятия и переключиться на использование нового программного продукта. В таком случае в дело вступают различные интеграционные механизмы, как то шины данных, rest-интерфейсы, разнообразные обмены тд.

Вместо того, чтобы описывать процессы «to be» на бумаге, происходит их проектирование с использованием реальных данных в установленной и настроенной системе.

Этап Моделирования позволяет гибко управлять бюджетом проекта, в конечном счете, выбор остается за заказчиком – согласится ли на типовые возможности «коробочного» продукта или же кастомизировать их согласно своим представлениям. Кроме того, на этом этапе можно скорректировать внедряемую функциональность, поняв, что от чего-либо второстепенного, с точки зрения проекта можно отказаться до лучших времен. Альтернативой может стать разбиение проекта на части – вместо полного перехода на новую информационную систему, часть функциональности могла бы остаться в старой системе, будучи интегрированной с новой информационной системой через шины данных, обмены и тд.

2.3 Разработка и устранение функциональных разрывов

Важным преимуществом «1С:Предприятия 8» является высокая скорость разработки прикладных решений, которое достигается благодаря применению концепции декларативного программирования, визуального редактирования, широко применяется идеология low-code. Для написания программного кода используется высокоуровневый предметно-ориентированный язык с понятийной моделью, максимально приближенной к задачам бизнеса (реализован подход Domain-Driven Design – предметно-ориентированное проектирование). На

данном этапе проекта производится эффективная и быстрая доработка программного продукта под требования, зафиксированные на первом этапе.

Важным аспектом данного этапа является следование методологии разработки, принятой в качестве стандарта, разработка тестов для ключевых функциональных контуров будущей системы, документирование сделанных изменений.

Положительным аспектом разработки отдельного Технического задания для реализация каждого зафиксированного изменения, является декомпозиция большого пласта работ на более мелкие части, за счет чего, достигается упрощение реализации и передачи в эксплуатацию.

Заключение

Следование успешным практикам во внедрении современных бизнес-систем становится существенным фактором достижения качества результата, соблюдения сроков проекта и управления бюджетом проекта.

Литература

1. 1С Акционерное общество. – Режим доступа: https://www.tadviser.ru/index.php/Компания:1С_Акционерное_общество. – (Дата обращения: 10.04.2023).
2. GAP-анализ: короткий путь от желаемого к действительному! – Режим доступа: https://4brain.ru/blog/gap-analiz-korotkij-put-ot-zhelaemogo-k-dejstvitelnomu/ . – (Дата обращения: 10.04.2023).

УДК 004.8

АНАЛИЗ И ПРОГНОЗИРОВАНИЕ ДИНАМИКИ COVID-19

О. В. Матыкина

Воронежский государственный университет

Введение

Пандемия COVID-19 привела к катастрофическим последствиям во всем мире. Вследствие этого возросли требования к обоснованности стратегических решений, принимаемых в области управления человеческими и экономическими ресурсами [1]. Для эффективного принятия решений необходимо иметь доступ к достоверной информации о динамике заболеваемости и прогнозах развития эпидемиологической ситуации. Модели машинного обучения могут помочь в моделировании эпидемиологических процессов и выделении закономерностей в данных. Для обучения таких моделей и разработки точных инструментов прогнозирования необходимо использовать обезличенные медицинские базы данных, а также выявить дополнительные факторы, влияющие на динамику эпидемиологического процесса. Тэлько комплексный подход к анализу данных и прогнозированию развития пандемии может обеспечить эффективное управление ресурсами и защиту населения от заболевания.

Цель данного исследования — научиться прогнозировать динамику заболеваемости COVID-19 в Воронежской области. В рамках исследования проводится разведочный анализ данных и разрабатывается инструмент среднесрочного прогнозирования динамики волн распространения эпидении. Описанные задачи реализованы на языке программирования Python с помощью различных его библиотек в интерактивной облачной среде Google Colab.

В предыдущем исследовании по данной теме [2] были проведены разведочный анализ и визуализация данных, отражающих результаты ПЦР тестирования на COVID-19 в Воронежской области, а также построена начальная модель прогнозирования динамики распространения COVID-19. В данной работе для построения среднесрочных прогнозов предполагается использование более сложной модели.

1. Исходные данные

В рамках исследования используются данные, предоставленные Воронежским областным клиническим консультативно-диагностическим центром (ВОККДЦ). Набор включает в себя данные обо всех ПЦР тестах на COVID-19, которые были проведены в Воронежской области в период с марта 2020 года по октябрь 2022 года. Датасет содержит следующие сведения о пациентах (рис. 1): идентификационный номер; пол; дата рождения; дата забора теста; результат теста; медицинская организация по месту жительства; медицинская организация, которая проводила тестирование; тест сдан амбулаторно или в стационаре; был ли тест сдан в одном из стационаров, в которые направляются преимущественно пациенты с тяжелыми случаями заболевания; зарегистрированы ли у данного пациента осложнения после перенесённого заболевания COVID-19; состоит ли пациент на диспансерном учете для реабилитации после COVID-19; была ли проведена вакцинация вторым компонентом более 2 недель назад; жив ли пациент. База данных пополняется. На момент 30 октября 2022 года она содержала сведения о результатах ПЦР тестирования более 1 миллиона пациентов.

ГУИД	Пол	Дата рождения.Начало года	Дата забора	Результат	Дата результата	МО по месту жительства	Проводящая исследование МО	Направлен	Стационар тяжёлый	Осложнения после ковид	Прошел первый этап УД	Жив	Вторая вакцина более двух недель назад
4e56970a	Женский	01.01.1962	28.04.2022	Отр.	28.04.2022	БУЗ ВО "ВГП №	БУЗ ВО "ВГП № 3"	Амбулаторно	Нет	Нет	Нет	Да	Нет
948164d7	Мужской	01.01.1952	01.04.2022	Отр.	01.04.2022	БУЗ ВО "ВГКП	БУЗ ВО "ВГКП №	Амбулаторно	Нет	Нет	Нет	Да	Нет
c3ad0a27	Женский	01.01.1936	24.04.2022	не обнаружен	25.04.2022	БУЗ ВО "ВГКП	БУЗ ВО "ВГКП №	Амбулаторно	Нет	Нет	Нет	Да	Нет
4d483606	Мужской	01.01.2008	14.04.2022	не обнаружен	15.04.2022	БУЗ ВО	БУЗ ВО "ВГКБСМП	Стационарно	Нет	Нет	Нет	Да	Нет
f54e3ea4	Женский	01.01.1956	12.04.2022	Отр.	12.04.2022	БУЗ ВО "ВГКП	БУЗ ВО "ВГКП №	Амбулаторно	Нет	Нет	Нет	Да	Нет
9b9b156f	Женский	01.01.1957	05.04.2022	Отр.	06.04.2022 9:16:30	БУЗ ВО	БУЗ ВО	Амбулаторно	Нет	Нет	Да	Да	Нет
46618c66	Женский	01.01.1967	04.04.2022	не обнаружен	05.04.2022	БУЗ ВО "ВГП №	БУЗ ВО "ВГП №	Амбулаторно	Нет	Нет	Нет	Да	Да
46618c66	Женский	01.01.1967	04.04.2022	Отр.	05.04.2022	БУЗ ВО "ВГП №	БУЗ ВО "ВГП №	Амбулаторно	Нет	Нет	Нет	Да	Да
31e0136b	Женский	01.01.1945	26.04.2022	Отр.	26.04.2022	БУЗ ВО "ВГКП	БУЗ ВО "ВГКП №	Амбулаторно	Нет	Нет	Нет	Да	Нет
835bcbe5	Женский	01.01.1961	08.04.2022	Отр.	08.04.2022	БУЗ ВО "ВГКБ	БУЗ ВО "ВГКБ №5"	Амбулаторно	Нет	Нет	Нет	Да	Нет
be78e808	Женский	01.01.2011	11.04.2022	не	11.04.2022	БУЗ ВО "ВГП №	АУЗ ВО "ВОККДЦ"	Амбулаторно	Нет	Нет	Нет	Да	Нет
be78e808	Женский	01.01.2011	11.04.2022	Отр.	11.04.2022 9:25:43	БУЗ ВО "ВГП №	БУЗ ВО "ВГП №	Амбулаторно	Нет	Нет	Нет	Да	Нет
be78e80f	Женский	01.01.1977	16.04.2022	обнаружена	19.04.2022	БУЗ ВО "ВГКБ	АУЗ ВО "ВОККДЦ"	Амбулаторно	Нет	Нет	Нет	Да	Нет
be78e80f	Женский	01.01.1977	26.04.2022	Отр.	26.04.2022	БУЗ ВО "ВГБ	БУЗ ВО "ВГБ №16"	Амбулаторно	Нет	Нет	Нет	Да	Нет
be78e80f	Женский	01.01.1977	21.04.2022	Отр.	21.04.2022	БУЗ ВО "ВГКБ	БУЗ ВО "ВГКБ	Амбулаторно	Нет	Нет	Нет	Да	Да
be78e80f	Женский	01.01.1977	19.04.2022	Отр.	19.04.2022	БУЗ ВО "ВГКБ	БУЗ ВО "ВГКБ	Амбулаторно	Нет	Нет	Нет	Да	Да
be78e80f	Женский	01.01.1977	16.04.2022	Отр.	16.04.2022	БУЗ ВО "ВГБ	БУЗ ВО "ВГБ №16"	Амбулаторно	Нет	Нет	Нет	Да	Нет
593c71bb	Женский	01.01.1948	28.04.2022	Пол.	29.04.2022	БУЗ ВО "ВГП №	БУЗ ВО "ВГП №	Амбулаторно	Нет	Нет	Нет	Да	Да
593c71be	Женский	01.01.1943	12.04.2022	Пол.	13.04.2022	БУЗ ВО "ВГП №	БУЗ ВО "ВГП №	Амбулаторно	Нет	Нет	Нет	Да	Да
593c71ca	Женский	01.01.1941	08.04.2022	Отр.	08.04.2022	БУЗ ВО "ВГКБ	БУЗ ВО "ВГКБ №5"	Стационарно	Нет	Нет	Нет	Да	Да
593c71ca	Женский	01.01.1941	01.04.2022	Отр.	01.04.2022	БУЗ ВО "ВГКБ	БУЗ ВО "ВГКБ №5"	Стационарно	Нет	Нет	Нет	Да	Нет
7a29837d	Женский	01.01.1964	18.04.2022	Отр.	18.04.2022	БУЗ ВО "ВГКБ	БУЗ ВО "ВГКБ	Амбулаторно	Нет	Нет	Нет	Да	Да
5ee763a3	Женский	01.01.1972	18.04.2022	Отр.	18.04.2022	БУЗ ВО "ВГП №	БУЗ ВО "ВГП №	Амбулаторно	Нет	Да	Да	Да	Да
5ee763a3	Женский	01.01.1972	04.04.2022	Отр.	04.04.2022	БУЗ ВО "ВГП №	БУЗ ВО "ВГП №	Амбулаторно	Нет	Да	Да	Да	Да

Рис. 1. Фрагмент исходных данных

2. Группировка данных

Для того, чтобы сократить количество строк и сделать данные более удобными для работы, была проведена группировка записей о ПЦР тестах по пациенту. Полученный датасет содержит следующие признаки (рис. 2): идентификационный номер пациента; пол; возраст; количество заболеваний; продолжительность заболеваний; период между заболеваниями; список дат начала заболевания; список дней болезни.

	id	Пол	Возраст	Кол-во заболеваний	Продолжительность заболеваний	Период между заболеваниями	Дата начала заболевания	Дни болезни
0	455f7acf-8c7a-11ea-969e-00155d0b0237	Женский	73	3	{20: 1, 8: 1, 11: 1}	{171: 1, 161: 1}	{Timestamp('2020-09-16 00:00:00'): 1, Timestam...	{numpy.datetime64('2020-09-16T00:00:00.0000000...
9	970a2709-db8e-11ea-96a3-00155d0b014f	Женский	49	3	{15: 1, 16: 1}	{127: 1, 172: 1}	{Timestamp('2020-08-11 00:00:00'): 1, Timestam...	{numpy.datetime64('2020-08-11T00:00:00.0000000...
1	973740d8-974a-11ea-969f-00155d0b0237	Женский	34	3	{12: 1, 14: 1}	{246: 1, 208: 1}	{Timestamp('2020-05-16 00:00:00'): 1, Timestam...	{numpy.datetime64('2020-05-16T00:00:00.0000000...
16	174f07a8-3e10-11eb-96a6-00155d0b0156	Женский	50	3	{11: 1, 3: 1}	{170: 1, 147: 1}	{Timestamp('2020-12-14 00:00:00'): 1, Timestam...	{numpy.datetime64('2020-12-14T00:00:00.0000000...
14	c8503ae2-1a7a-11eb-96a5-00155d0b0156	Женский	42	3	{32: 1, 15: 1}	{192: 1, 119: 1}	{Timestamp('2020-10-30 00:00:00'): 1, Timestam...	{numpy.datetime64('2020-10-30T00:00:00.0000000...

Рис. 2. Фрагмент сгруппированных данных

Группировка позволила существенно сократить объем данных без потери качества, и даже обнаружить некоторые закономерности на этапе разведочного анализа данных.

3. Разведочный анализ данных

На данном этапе необходимо было узнать, есть ли у заболевших какие-либо половозрастные особенности. Для этого были построены гистограммы распределения по полу всех людей, сдававших тесты, и людей, у которых результат теста был положительным. Гистограммы представлены на рис. 3, 4.

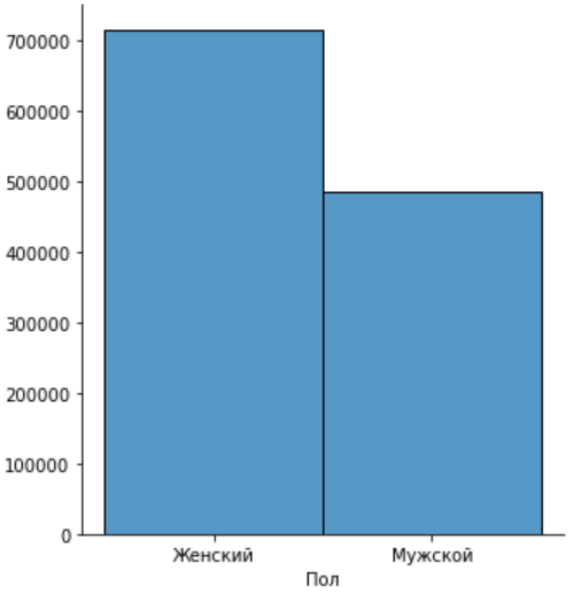


Рис. 3. Распределение по полу всех сдававших тесты

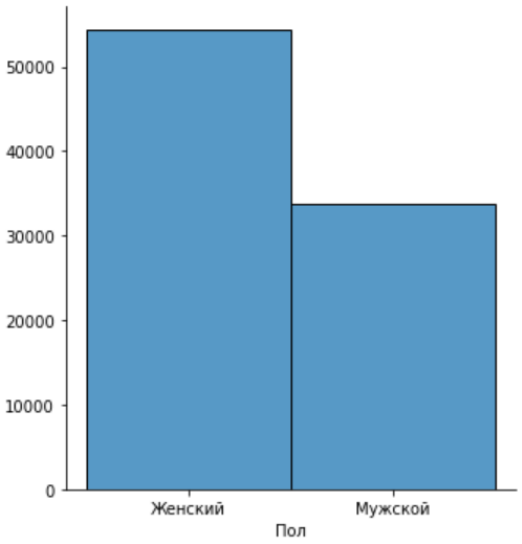


Рис. 4. Распределение по полу людей с положительным результатом

По гистораммам можно сделать вывод о том, что большее количество заболевших женщин объясняется тем, что больше женщин сдавало ПЦР тест.

Далее была построена гистограмма зависимости количества заболевших от пола и возраста. Она представлена на рис. 5.

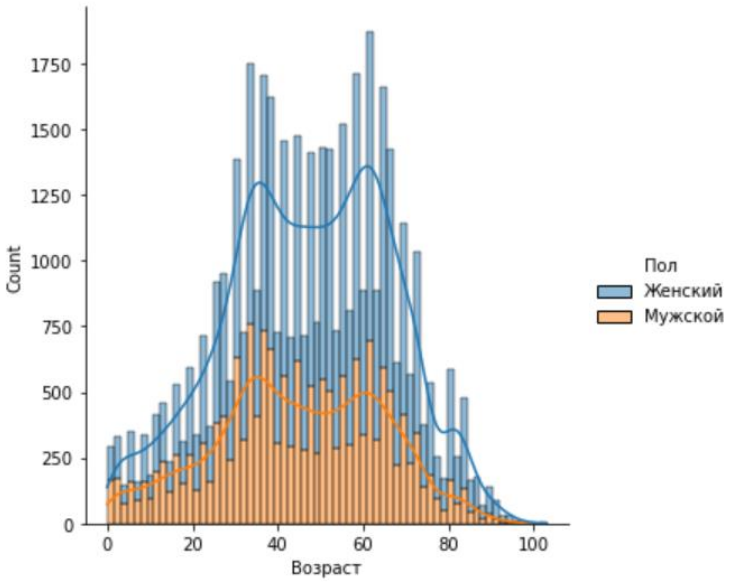


Рис. 5. Распределение по полу и возрасту заболевших

По гистограмме видно, что распределение заболевших имеет два пика: в возрасте 35 и 65 лет. Гистограмма зависимости повторно заболевших от пола и возраста (рис. 6) имеет похожее распределение, однако второй пик сдвигается немного влево — на возраст около 60 лет.

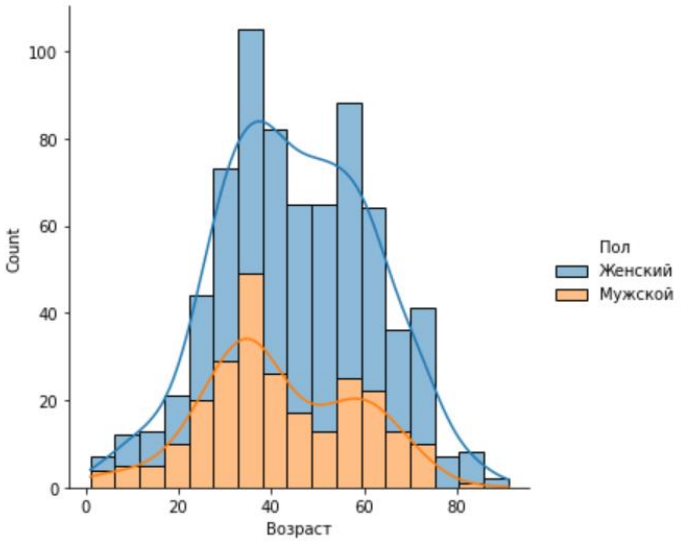


Рис. 6. Распределение по полу и возрасту повторно заболевших

4. Базовое решение

Для построения моделей предсказания количества заболевших данные были преобразованы во временной ряд и сглажены семидневной скользящей. График сглаженного временного ряда представлен на рис. 7.

Для обучения и тестирования моделей данные были разбиты на обучающую и тестовую выборку. Так как данные представляют собой временной ряд, они не перемешивались, а в качестве тестовой выборки были взяты последние 20 суток всего периода наблюдений.

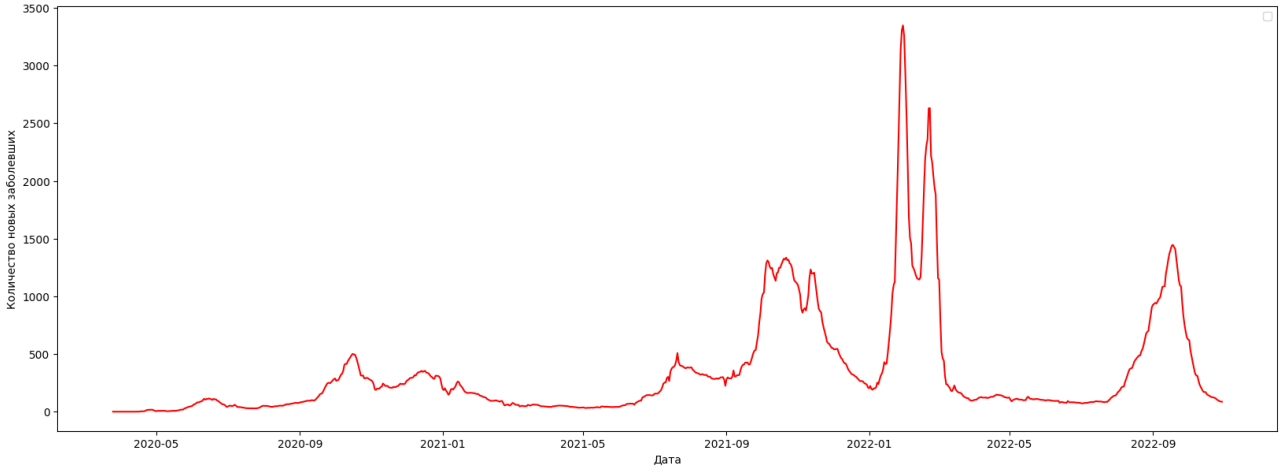


Рис. 7. График количества заболевших

В качестве базовой модели была взята модель SARIMAX [3] с параметрами order = (1, 1, 1), seasonal_order = (1, 0, 0, 2). На рис. 8 представлены реального и предсказанного количества заболевших.

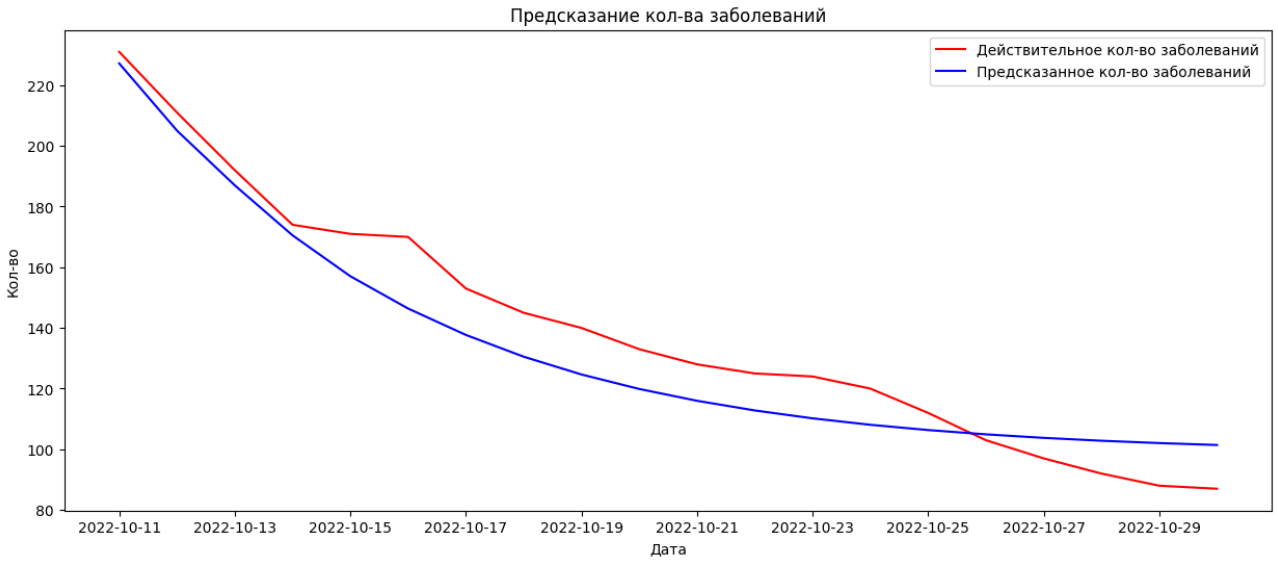


Рис. 8. Результат работы базовой модели

5. Гибридная модель глубокого обучения

В качестве основного подхода разработана гибридная модель глубокого обучения [4-5], которая включает рекуррентные сети долгой краткосрочной памяти (LSTM), при этом признаки из временных рядов автоматически извлекаются с помощью сверточной нейронной сети (CNN).

Особенностью архитектуры рекуррентных нейронных сетей является наличие обратных связей, что позволяет моделировать процесс запоминания элементов списков и последовательностей. Сети архитектуры LSTM обладают долгосрочной и краткосрочной памятью. Схема основного блока нейронной сети LSTM приведена на рис. 9. Архитектура сети LSTM включает несколько таких блоков.

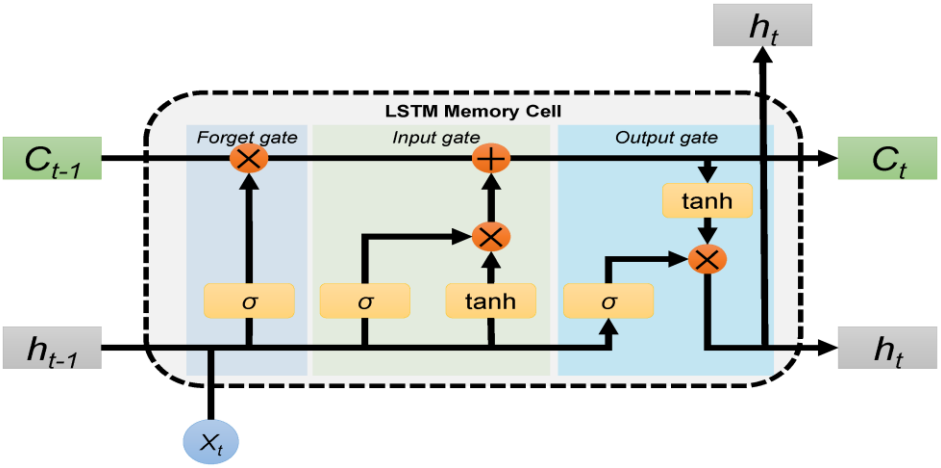


Рис. 9. Схема ячейки LSTM

Сети LSTM успешно справляются с задачами обработки и прогнозирования временных рядов, поскольку между важными событиями во временном ряду могут быть задержки неограниченной продолжительности, что обязывает использовать для предсказания данные из далекого прошлого.

Цель использования моделей CNN и LSTM в прогнозировании временных рядов заключается в том, что LSTM-модели способны эффективно обрабатывать последовательности, а CNN-модели могут фильтровать шум и выявлять важные признаки. Использование этих моделей в нейронной сети глубокого обучения может улучшить результаты прогнозирования. Архитектура сети CNN-LSTM изображена на рис. 10.

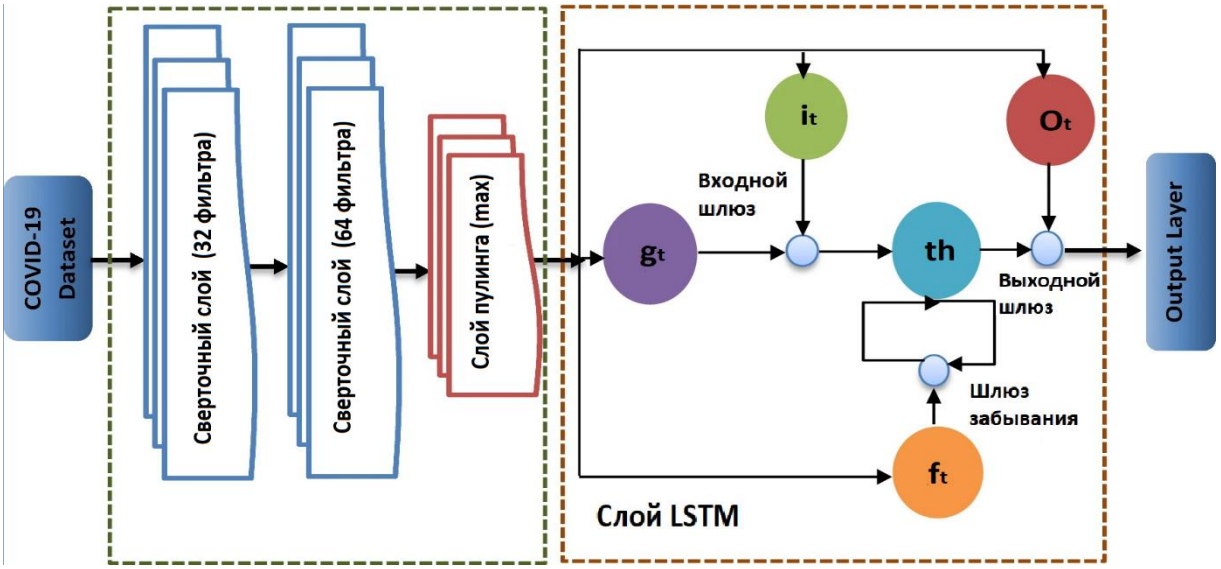


Рис. 10. Архитектура сети CNN-LSTM

Использование keras-tuner позволило подобрать оптимальные гиперпараметры сети. На рис. 11 представлен график реального и предсказанного количества заболевших.

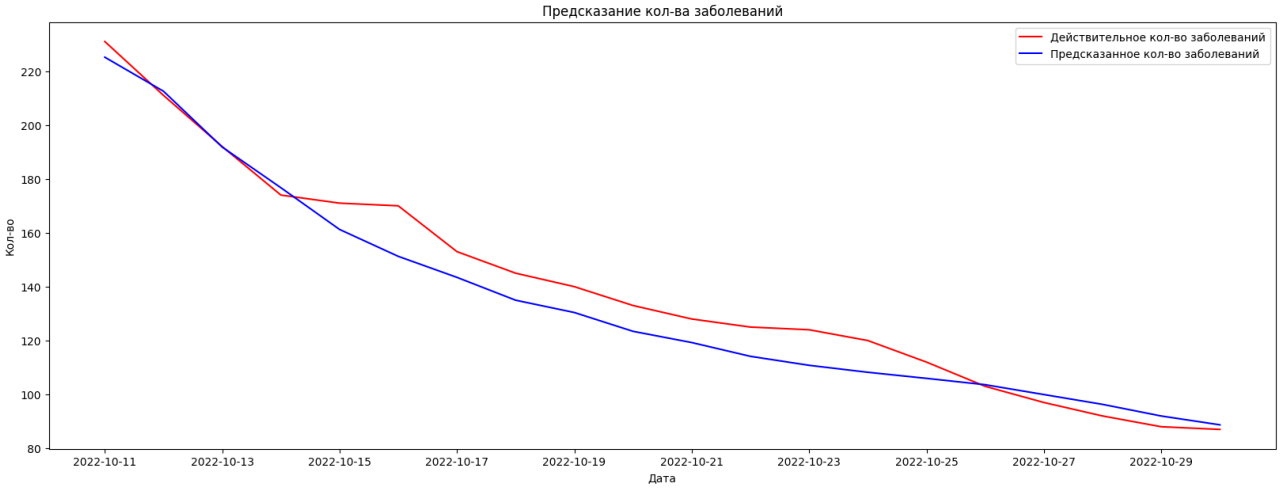


Рис. 11. Результат работы гибридной модели

6. Оценка моделей

В качестве основной метрики был выбран коэффициент детерминации. Дополнительно были рассчитаны корень из средней квадратичной ошибки и средняя абсолютная ошибка в процентах. Результаты построенных моделей приведены в табл. 1. Лучший результат показала модель CNN-LSTM.

Таблица 1

Точность построенных моделей

Модель	R^2	RMSE	MAPE
SARIMAX	0.909123	12.082673	8.516641
CNN-LSTM	0.954656	8.534876	5.251771

Заключение

В ходе данного исследования был проведен разведочный анализ исходных данных, построена модель машинного обучения на основе нейронных сетей для прогнозирования количества заболевших и проведено сравнение со статистической моделью SARIMAX.

В результате построения соответствующей архитектуры и подбора гиперпараметров модели лучше всего показала себя модель CNN-LSTM с коэффициентом детерминации 0.95.

Литература

1. World Bank. Europe and Central Asia Economic Update, Fall 2020: COVID-19 and Human Capital. Washington, DC: World Bank, 2020. – Режим доступа: https://openknowledge.worldbank.org/handle/10986/34518 (Дата обращения: 22.04.2023)

2. Каширина, И. Л. Анализ, моделирование и прогнозирование COVID-19 на основе данных Воронежской области / И. Л. Каширина, Д. О. Ершов // Актуальные проблемы прикладной математики, информатики и механики : Сборник трудов Международной научной конференции, Воронеж, 13–15 декабря 2021 года. – Воронеж, 2022. – С. 174-180.

3. Документация модели SARIMAX из библиотеки statsmodels – Режим доступа: https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html (Дата обращения: 22.04.2023)

4. Инструментальные методы оценки человеческого капитала : Теория и прикладные аспекты / Н. В. Яковенко, Т. В. Азарнова, И. Л. Каширина [и др.]. – Воронеж : Издательство "Цифровая полиграфия", 2022. – 177 с.

5. Ketu, S. India perspective: CNN-LSTM hybrid deep learning model-based COVID-19 prediction and current status of medical resource availability. / S. Ketu, P.K. Mishra // Soft Comput. – 2022. – № 26. – С. 645–664.

УДК 004.8

СРАВНЕНИЕ РЕЗУЛЬТАТОВ РЕШЕНИЙ ЗАДАЧИ РАСПОЗНАВАНИЯ ТЕКСТА С ПОМОЩЬЮ МОБИЛЬНОГО УСТРОЙСТВА

А. Е. Мащенко, С. Ю. Болотова

Воронежский государственный университет

Введение

В работе [1] были предложены и рассмотрены методы распознавания текстовой информации с помощью мобильного устройства, работающего под управлением операционной системы iOS, на примере задачи распознавания ценника. При решении задачи распознавания было выделено две подзадачи: определение на изображении границ прямоугольной области - ценника и выделение границы символа на изображении и его последующей его обработкой с применением методов машинного обучения. В [1] реализовано три механизма определения прямоугольной границы ценника:

1) средствами iOS с помощью встроенных возможностей фреймворка Vision для определения прямоугольных областей;
2) с помощью модели машинного обучения, обученной на нахождение прямоугольных областей на изображении;
3) с помощью ручной настройки границы, в рамках которой будет проводиться распознавание текста.

В данной работе приводятся сравнительный анализ рассмотренных способов распознавания.

1. Сравнительный анализ

Для сравнительного анализа реализованных методов распознавания ценника был проведен ряд экспериментов на устройстве iPhone 11 с процессором Apple A13 Bionic.

В качестве объекта распознавания созданы и распечатаны сто ценников, удовлетворяющие принятым условиям к расположению элементов внутри ценника: название товара написано наверху в пределах 30 процентов от общей высоты, цена товара - в правом нижнем углу.

Для проверки качества распознавания эксперименты были проведены при различных внешних факторах: нормальные условия (цвет фона вокруг ценника светлый, освещение дневное), условия повышенной освещенности (цвет фона контрастный, используется дополнительное освещение), неблагоприятные внешние условия (цвет фона сливается с фоном ценника, недостаточная освещенность, часть ценника перекрыта).

При проведении экспериментов фиксировались значения следующих метрик:

- время распознавания прямоугольной области на изображении;
- время распознавания символов на изображении;
- время обработки изображений символов с помощью модели машинного обучения;
- время вынесения предсказания;
- время синтаксического анализа;
- время одного цикла распознавания;
- общее время распознавания информации на ценнике;

- количество необходимых для распознавания циклов.

На рис. 1 графически представлено расположение вышеперечисленных этапов работы алгоритма относительно друг друга.

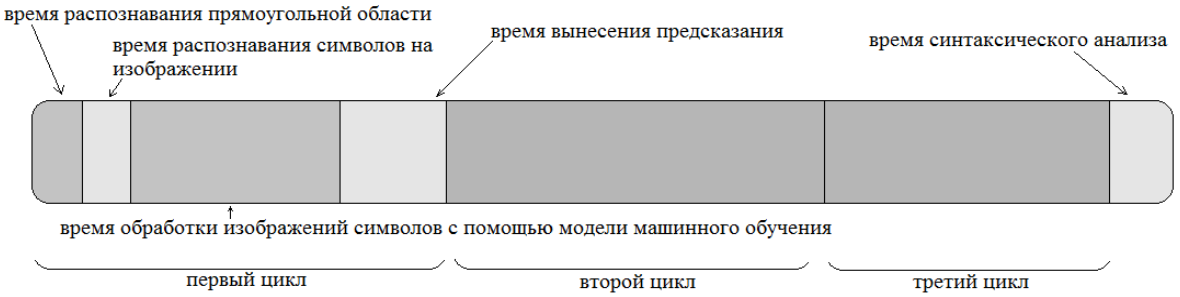


Рис. 1. Порядок следования этапов работы алгоритма

Фрагмент полученных результатов экспериментов приведен в табл. 1 и 2. В рамках эксперимента каждый из тестовых ценников был идентифицирован тремя указанными выше способами распознавания, количество циклов в каждом случае варьировалось в зависимости от результатов работы реализованного в мобильном приложении компонента, ответственного за вынесение предсказания.

Таблица 1

Результаты экспериментов при нормальных условиях

	Среднее время обработки изображений с помощью методов машинного обучения (с)	Среднее время одного цикла распознавания (с)	Среднее общее время распознавания (с)	Среднее количество необходимых для распознавания циклов (шт)
Средствами iOS	0,769921738	0,885643058	11,92992133	9,8
С помощью сторонней модели	0,758965412	0,845984526	11,16549845	9,12
С помощью ручной настройки	0,668920404	0,705863277	5,67731607	5,93

Таблица 2

Результаты эксперимента в условиях повышенной освещенности

	Среднее время обработки изображений с помощью методов машинного обучения (с)	Среднее время одного цикла распознавания (с)	Среднее общее время распознавания (с)	Среднее количество необходимых для распознавания циклов (шт)
Средствами iOS	0,750154949	0,86782183	10,7815698	8,96

			7	
С помощью сторонней модели	0,795959169	0,88458965	11,35946859	9,23
С помощью ручной настройки	0,969108011	1,024160973	8,963348667	6,43

В таблицах не приведены результаты опытов для некоторых из вышеперечисленных метрик, так как они представляют собой близкие значения одного порядка и оказывают крайне малое влияние на скорость работы алгоритма.

На основании полученных результатов был проведен анализ и получены следующие выводы. Метод для определения информации на ценнике, где граница ценника определяется средствами платформы iOS с помощью фреймворка Vision практически не отличается по характеристикам от метода, где граница ценника определяется с помощью сторонней модели машинного обучения. Это объясняется сходством внутренней реализации данных методов. При обоих условиях метод определения информации на ценнике, где пользователь может сам настроить границу ценника, дает наиболее быстрые результаты, что обусловлено более точным определением границы ценника. При этом среднее количество циклов в одной сессии распознавания у этого метода меньше благодаря более точному определению границ ценника. Метод, использующий средства системы iOS, дает лучшие результаты работы при условиях повышенной освещенности. Данный факт можно объяснить тем, что границу ценника при таких условиях определить легче, что приводит к дальнейшему сокращению необходимой к выполнению работы и уменьшению как общего времени выполнения, так и числа необходимых циклов. У «ручного» же метода время распознавания при условии повышенной освещенности, наоборот, возрастает, поскольку при таких условиях в качестве символов текста распознаются различные области на изображении, которые ими не являются, к примеру, области штрих-кода. Это приводит к дальнейшему возрастанию вычислительной нагрузки и увеличению необходимой к обработке информации. Подтверждается данная гипотеза и результатами опытов, где можно увидеть крайне высокое значение среднего времени обработки изображений с помощью методов машинного обучения, которое превышает аналогичные показатели у других методов.

Заключение

Все рассмотренные в работе методы решения задачи распознавания текста с помощью мобильного устройства позволяют достичь высокой точности распознавания. Проведен ряд экспериментов в условиях недостаточной, нормальной и повышенной освещенности. В первом случае точность составила 88%, в последних - около 93%. Точность распознавания символов непосредственно зависит от точности распознавания границ прямоугольника, в котором они располагаются. Точное определение границ также приводит к дальнейшему сокращению и общего времени выполнения.

Литература

1. Мащенко, А. Е. Решение задачи распознавания текста с помощью мобильного приложения / А. Е. Мащенко, С. Ю. Болотова // Актуальные проблемы прикладной математики, информатики и механики : сборник трудов Международной научной конференции, Воронеж, 13-15 декабря 2021 г. Воронеж, 2022. С. 200-202. ISBN 978-5-6045486-6-0.
2. Плас Дж. В. Python для сложных задач: наука о данных и машинное обучение / Дж.

В. Плас. – СПб. : Питер, 2018. – 576 с.
3. Tam A. Machine Learning by Tutorials (Second Edition): Beginning Machine Learning for Apple and iOS / Audrey Tam, Matthijs Hollemans, Alexis Gallagher and Chris LaPollo. – Razeware LLC, 2020. – 600 p.
4. Джулли А. Библиотека Keras – инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и Tensorflow / А. Джулли, С. Пал. – М. : ДМК Пресс, 2018. – 294 с.
5. Thakkar M. Beginning Machine Learning in iOS: CoreML Framework / Mohit Thakkar Vadodara. – Apress, Berkeley, CA, 2019. – 157 p.

УДК 004.9

СОЗДАНИЕ МОДУЛЯ ВЗАИМОДЕЙСТВИЯ МЕЖДУ АБИТУРИЕНТОМ И ПРИЕМНОЙ КОМИССИЕЙ В ПРИЛОЖЕНИИ «СИСТЕМА ДЛЯ РАБОТЫ С АБИТУРИЕНТАМИ»

Н. Р. Мишин

Воронежский государственный университет

Введение

Постоянно развивающиеся технологии оказывают огромное влияние на различные сферы жизни человека. Одной из таких сфер является образование. В современном мире возрастает интерес к образованию, следовательно, количество людей, желающих поступить в высшие учебные заведения, становится все больше. В связи с этим важно обеспечить максимально удобные и эффективные условия для взаимодействия с будущими студентами.

Конечная цель любой образовательной организации – обеспечить наилучшие условия для студентов и обеспечить высокое качество обучения. Однако, кроме традиционных задач, таких как разработка программы обучения и оценка успеваемости студентов, важную роль играет эффективное взаимодействие с абитуриентами в период подачи документов. В этот период, приемная комиссия выполняет ряд задач, такие как: прием и обработка документов, организация вступительных испытаний и многое другое. В свою очередь, абитуриентам нужно связаться с университетом, чтобы получить необходимую информацию и подать документы.

Одним из способов улучшения взаимодействия является использование информационных технологий. Благодаря этому можно значительно сократить время на обработку документов и организацию вступительных испытаний, что позволит приемной комиссии более эффективно работать с абитуриентами. В результате, студенты получат более быстрый и качественный процесс приема в университет. В этом контексте был разработан модуль, который улучшает взаимодействие между абитуриентами и приемной комиссией, обеспечивая удобный и быстрый обмен информацией.

1. Инструменты реализации

Для реализации системы использованы следующие инструменты:

- Java – язык программирования. Обладает высокой производительностью и надежностью, что делает его отличным выбором для создания сложных и масштабируемых систем;
- Spring Framework (Boot, Security) – фреймворк для создания масштабируемых и надежных приложений на Java. Предоставляет удобный инструментарий для реализации шаблона проектирования MVC (Model-View-Controller), что позволяет разделить логику приложения на компоненты;
- PostgreSQL – система управления реляционными базами данных. Она предоставляет широкий набор возможностей и функций, включая поддержку транзакций, целостность данных, масштабируемость и высокую производительность.

2. Разработка модуля

На данном этапе необходимо создать модуль взаимодействия между приемной комиссией и абитуриентом. Для достижения этой цели необходимо решить следующие задачи:

1. Предоставление абитуриенту возможности подать заявление на поступление в учебное заведение;
2. Обеспечить оперативное взаимодействие между абитуриентом и приемной комиссией;
3. Предоставить доступ к информации о поступлении;
4. Обеспечить возможность отслеживания статуса заявления на поступление.

Процесс создания модуля разбит на следующие шаги:

1. Создание базы данных;
2. Определение сущностей;
3. Создание модели данных;
4. Создание контроллеров;
5. Создание представлений;
6. Создание сервисов;
7. Интеграция компонентов.

Будущий студент, прошедший регистрацию в системе, должен заполнить контактную информацию о себе, а также прикрепить документы, необходимые для поступления (рис. 1).

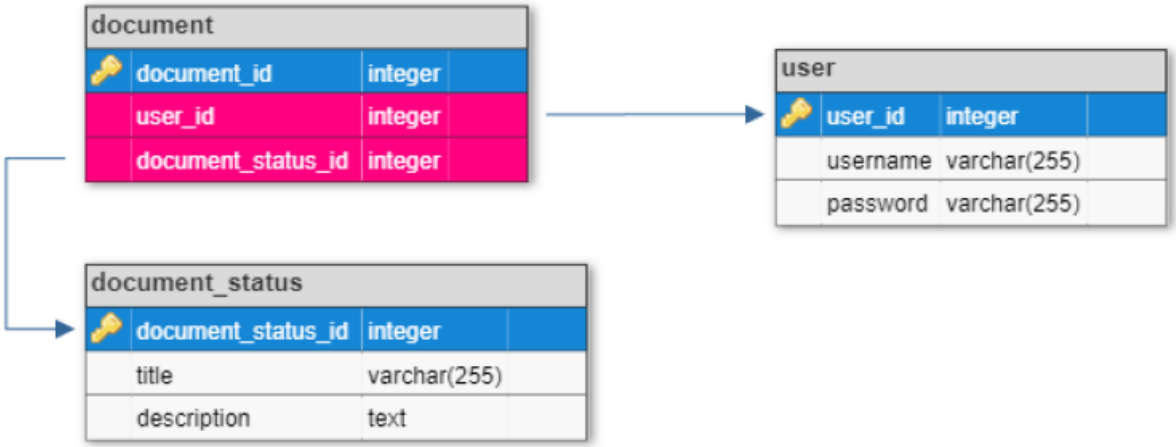


Рис. 1. Модель прикрепляемых документов

После прохождения основной процедуры заполнения профиля, автоматически формируется запрос в приемную комиссию на проверку соответствия прикрепленных документов. Если абитуриент заполнил всё верно, то он получает право на выбор факультета и направления обучения (рис. 2).

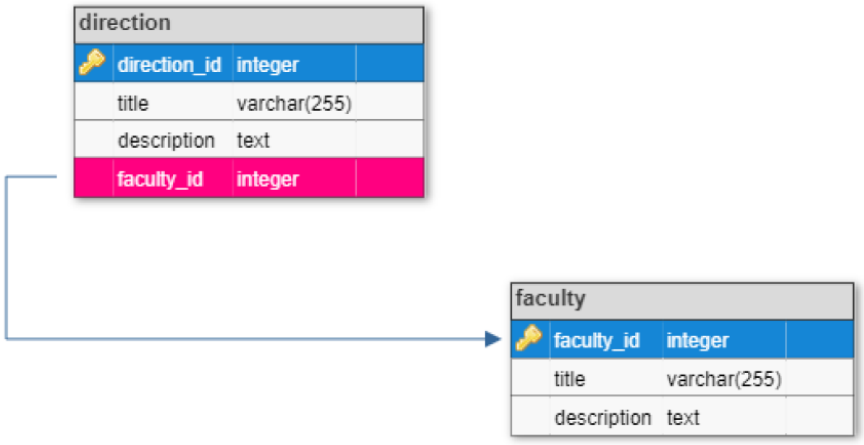


Рис. 2. Модель факультет-направление

Для этого, будущему студенту необходимо самостоятельно выбрать интересующее направление и отправить запрос в приемную комиссию, для участия в конкурсе по вузу.

В случае, если запрос требует уточнения или дополнительной информации, приемная комиссия должна связаться с абитуриентом через модуль обратной связи и запросить необходимые данные. После получения дополнительной информации, приемная комиссия принимает решение о приеме абитуриента на выбранное направление и отправляет уведомление об этом.

Стоит отметить, что обеспечение безопасности взаимодействия между студентом и приемной комиссией – это крайне важный аспект, который необходимо учитывать при разработке и эксплуатации системы.

Основная мера безопасности, выбранная при создании модуля – аутентификация и авторизация (рис. 3).

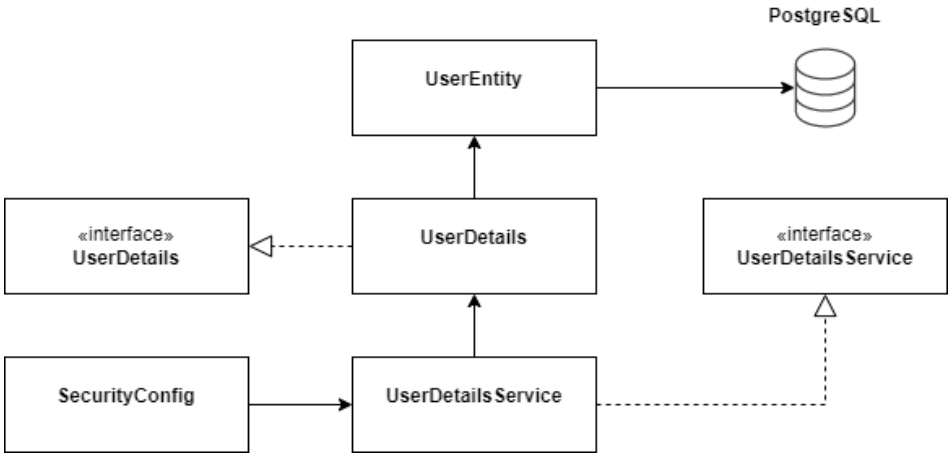


Рис. 3. Система авторизации и аутентификации

Система обеспечивает проверку подлинности и авторизации пользователей, чтобы гарантировать, что только правильные пользователи имеют доступ к системе и ее данным. Для этого используется метод аутентификации, такой как логины и пароли. Также была введена ролевая модель для зарегистрированных пользователей (рис. 4), чтобы контролировать доступ к различным частям приложения.

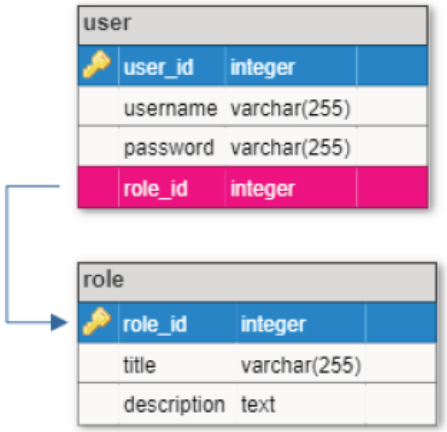


Рис. 4. Ролевая модель пользователей

Для предоставления актуальной информации и улучшения коммуникации между студентами и университетом, была разработана новостная лента (рис. 5).

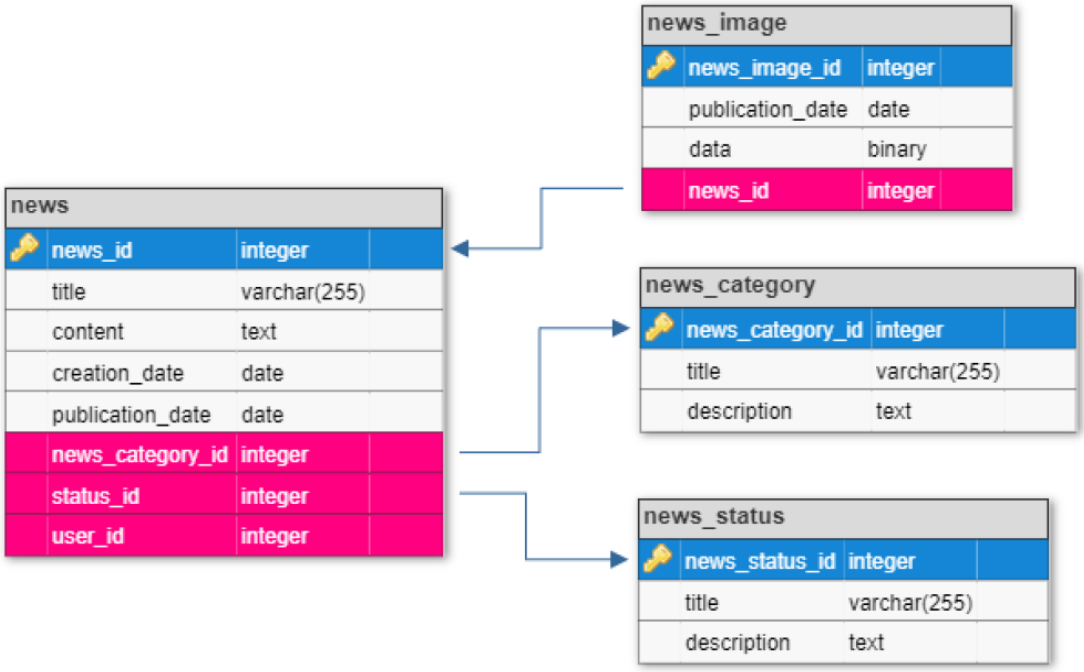


Рис. 5. Модель новостной ленты

Новостная лента представляет собой список сообщений, которые могут содержать различную информацию, связанную с процессом поступления, учебной деятельностью и другими событиями, происходящими в учебном заведении.

Таким образом, система позволяет абитуриентам отправлять запросы в приемную комиссию и получать актуальную информацию из новостной ленты, приемной комиссии – обрабатывать запросы и отправлять уведомления студентам.

Конечная версия модуля включает в себя следующие компоненты (рис. 6):

1. Новостная лента, на которой размещаются все последние новости, связанные с жизнью университета, результаты вступительных испытаний и т.д.
2. Форма обратной связи, где абитуриенты могут задавать вопросы приемной комиссии, получать консультации и уточнять информацию о приемной кампании.
3. Форма подачи документов, где абитуриенты могут загружать необходимые документы для поступления в университет, в том числе сканы документов и прочие материалы.
4. Форма регистрации абитуриентов.

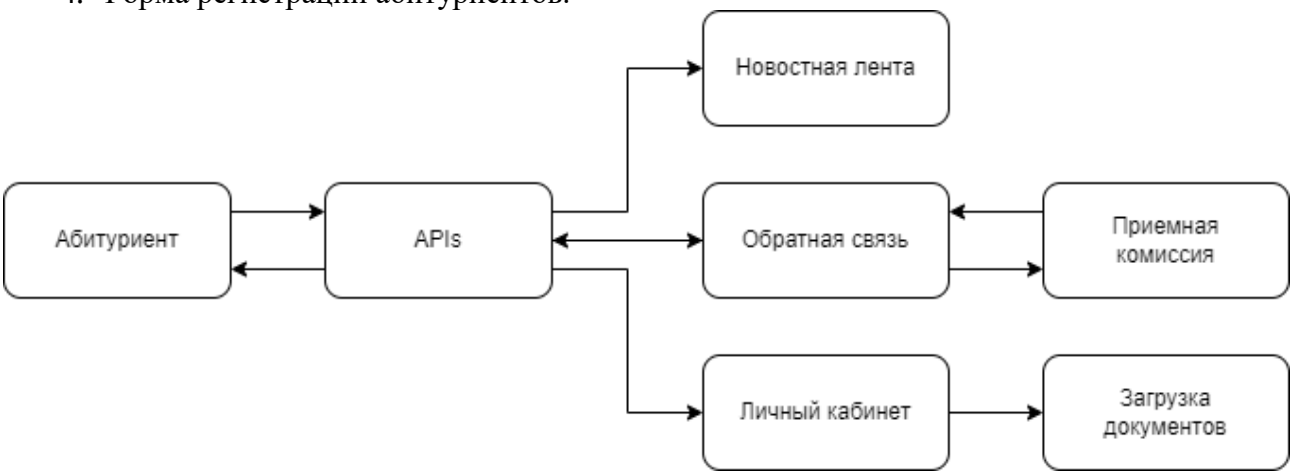


Рис. 6. Компоненты разрабатываемого модуля

Заключение

В заключении можно сказать, что разработанный модуль для взаимодействия с абитуриентами и приемной комиссией в приложении «Система для работы с абитуриентами» является важным инструментом в улучшении работы университета. Модуль позволяет значительно упростить процесс приема документов от будущих студентов, а также обеспечивает безопасность хранения персональных данных, благодаря механизмам авторизации и аутентификации. Новостная лента является важной функциональной возможностью, позволяющая уведомить абитуриентов о происходящих событиях. Также стоит отметить, что разработанная система может быть использована для работы с уже зачисленными студентами. В целом, разработка системы, позволяет сделать поступление в университет простым процессом.

Литература

1. Крейг, У. Spring in Action / У. Крейг. — 6-е изд. — Москва : Manning Publications, 2022. — 546 с.

2. Гутьеррес, Ф. Spring Boot 2. Лучшие практики для профессионалов / Ф. Гутьеррес. — : Питер, 2020. — 464 с.

3. Christian, B. Java Persistence with Spring Data and Hibernate / B. Christian, K. Gavin, G. Gary. — : Manning Publications, 2022. — 616 с.

УДК 519.61

О ФАКТОРИЗАЦИИ КВАДРАТИЧНОГО МАТРИЧНОГО ПУЧКА

А.А.Мозговая

Воронежский государственный университет

Введение

Системы линейных дифференциальных уравнений второго порядка

$$\ddot{x}(t) + B\dot{x}(t) + Cx(t) = f(t)$$

возникают во многих приложениях. Роль характеристических функций для них играют квадратичные пучки

$$\lambda \mapsto \lambda^2 I + \lambda B + C.$$

Исследование пучка и соответствующего уравнения существенно упрощается, если пучок удается факторизовать, т.е. представить в виде произведения

$$\lambda^2 I + \lambda B + C = (\lambda I - Z)(\lambda I - X),$$

поскольку это приводит к уменьшению в два раза размерностей матриц, с которыми приходится работать. Построение факторизации сводится к нахождению корня X характеристического уравнения

$$AX^2 + BX + C = 0.$$

Большинство авторов для нахождения корня X использует различные способы решения нелинейных уравнений (метод Ньютона [7], метод последовательных приближений [2] и др.). Для нахождения корня в настоящей работе применяется алгоритм, который мы не встречали в литературе. Но его принципиальная идея не является новой. Она основан на методе доказательства существования корня, изложенном в [6]. Преимущество этого алгоритма заключается в том, что он позволяет найти все корни X , что дает возможность сравнивать их друг с другом.

Выбор корня определяется набором порождающих его собственных значений так называемой сопровождающей матрицы. В данной статье приведено сравнение различных простых методов выбора собственных значений для поиска корня характеристического уравнения.

1. Алгоритм факторизации

Пусть $n \in \mathbb{N}$. Множество всех комплексных матриц размера $n \times n$ обозначим через $\mathbb{C}^{n\times n}$. Символ I обозначает единичную матрицу. Пусть $B, C \in \mathbb{C}^{n\times n}$. Рассмотрим *квадратичный матричный пучок*

$$\lambda \mapsto \lambda^2 I + \lambda B + C. \qquad (1)$$

Спектром пучка (1) называют множество всех $\lambda \in \square$, для которых матрица $\lambda^2 + \lambda B + C$ необратима. В настоящей работе мы ограничиваемся пучками с единичным старшим коэффициентом и самосопряженными матрицами A и B. Дифференциальные уравнения с самосопряженными коэффициентами особенно часто встречаются в приложениях. По поводу пучков с нетривиальными старшими коэффициентами см., например, [4].

Пучок (1) представляет собой характеристическую функцию дифференциального уравнения

$$\ddot{x}(t) + B\dot{x}(t) + Cx(t) = f(t), \ t \in \square.$$

Известно, что решение этого уравнения с начальными условиями

$$x(0) = u_0, \dot{x}(0) = u_1$$

представимо в виде

$$x(t) = \dot{U}(t)u_0 + U(t)(u_1 - Bu_0) + \int_0^t U(t-s) f(s) ds,$$

где

$$U(t) = \frac{1}{2\pi i} \int_\Gamma e^{\lambda t} (\lambda^2 I + \lambda B + C)^{-1} d\lambda,$$

а контур Γ окружает спектр пучка.

Факторизацией пучка (1) называют его разложение вида

$$\lambda^2 I + \lambda B + C = (\lambda I - Z)(\lambda I - X).$$

Правым корнем пучка (1) называют матрицу X, удовлетворяющую равенству

$$AX^2 + BX + C = 0.$$

Известно, что если X — правый корень, то пучок допускает факторизацию

$$\lambda I + \lambda B + C = (\lambda I + X + B)(\lambda 1 - X).$$

Матрицу $Z = -X - B$ называют в этом случае *левым корнем*. Говорят, что корни X и Z образуют *полную пару*, если матрица $X - Z$ обратима. Наличие факторизации упрощает вычислительную формулу для решения дифференциального уравнения: если X и Z образуют полную пару, то [5, с. 28], [6, с. 77, теорема 2.16] фундаментальное решение дифференциального уравнения представимо в виде

$$U(t) = \left[\frac{1}{2\pi i} \int_\Gamma e^{\lambda T} (\lambda I - X)^{-1} d\lambda - \frac{1}{2\pi i} \int_\Gamma e^{\lambda T} (\lambda I - Z)^{-1} d\lambda \right] (X - Z)^{-1}. \qquad (2)$$

Сопровождающей матрицей к пучку (1) называют блочную матрицу

$$C_1 = \begin{pmatrix} 0 & I \\ -C & -B \end{pmatrix}. \qquad (3)$$

Нахождение правого корня X сводится к нахождению инвариантного подпространства размерности n матрицы C_1.

Работа посвящена обсуждению следующего алгоритма нахождения правого корня пучка (и тем самым факторизации пучка). Выпишем жорданов базис для C_1. Возьмем какой-либо полный набор собственных векторов и отвечающих им присоединенных векторов. Их линейная оболочка образует инвариантное подпространство. Подберем собственные векторы так, чтобы общее число векторов равнялось n. Из выбранных векторов как из столбцов построим блочную матрицу

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix}. \tag{4}$$

Умножим этот блочный столбец справа на X_1^{-1}. Получится блочный столбец вида

$$\begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} X_1 X_1^{-1} \\ X_2 X_1^{-1} \end{pmatrix}.$$

Подчеркнем, что при таком умножении линейная оболочка столбцов не меняется. Матрица $X = X_1 X_1^{-1}$ оказывается правым корнем. После этого находится левый корень: $Z = -X - B$. Затем проверяется, является пара корней X и Z полной; для этого проверяется, является ли матрица $X - Z$ обратимой. В конце нетрудно выписать саму факторизацию.

2. Постановка задачи и ее решение

Даже если матрица $X - Z$ оказывается обратимой, полученная факторизация может быть не очень удобной. Действительно, с вычислительной точки зрения случай необратимой матрицы $X - Z$ мало отличается от случая плохо обратимой матрицы $X - Z$. Плохо обратимой матрицей $X - Z$ мы считаем матрицу, у которой *число обусловленности* [1, с. 264]

$$\kappa = \|X - Z\| \cdot \|(X - Z)^{-1}\|$$

является большим. Число обусловленности характеризует неизбежную потерю точности при работе с такой матрицей: например, если $\kappa = 1000$, то при каждом умножении на матрицу $X - Z$ или на матрицу $(X - Z)^{-1}$ (см. формулу (2)) точность (число достоверных значащих цифр) уменьшается в 1000 раз (в нашем случае пропадает 3 десятичных знака).

В то же время различных правых корней X и соответственно факторизаций много. Мы ставили свои численные эксперименты на матрицах состоящих из случайных чисел. У таких матриц кратные собственные значения и тем более присоединенные векторы возникают с нулевой вероятностью. Поэтому каждому набору из n собственных векторов сопровождающей матрицы (3), имеющей размер $2n \times 2n$, соответствует свой правый корень X. Таким образом, правых корней (и тем самым способов решения задачи о факторизации) в нашем случае имеется C_{2n}^n. Цель работы состояла в том, чтобы предложить сравнительно простое правило,

позволяющее найти если не оптимальный корень X, то хотя бы близкий к оптимальному (т.е. имеющий κ, близкое к минимально возможному). Для этого было предложено четыре правила, претендующих на близость к оптимальному:

- Матрица X строилась по первым n собственным значениям матрицы A, которые появлялись в процессе вычислений спектра. Кажется, что это случайный выбор, но это не совсем так: используемый для нахождения собственных значений QR-алгоритм, располагает собственные числа примерно в порядке убывания модулей собственных значений.
- Сначала собственные значения матрицы A сортировались в порядке возрастания действительных частей, а затем корень X строился по первым n.
- Собственные значения матрицы A сортировались в порядке возрастания мнимых частей, а корень X строился по первым n. Это правило не совсем аналогично предыдущему, поскольку в экспериментах участвовали матрицы A, состоящие из действительных чисел. В результате деление спектра на две части всегда проходило по действительной оси. Более того, как правило, часть собственных значений имела нулевые мнимые части и они были отсортированы между собой случайным образом.
- В одну часть включались собственные значения, наиболее близкие друг к другу. Для этого сначала выбиралась пара собственных значений, которые наиболее близки друг к другу. Затем к ним добавлялось собственное значение, которое наиболее близко к этой паре. Затем собственное значение, которое наиболее близко к уже имеющимся. И так до тех пор, пока не наберется n штук. По выбранным собственным значениям строился корень X.

Для случайно сформированных матриц A и B перебирались все C_{2n}^{n} матриц (4), для каждой из них строился корень X и вычислялось число обусловленности. Затем выбиралось наименьшее из чисел обусловленности. Затем число обусловленности вычислялось для каждого из предлагаемых правил и сравнивалось с наименьшим возможным. Результаты приведены в таблице 1. Конечно, перебор всех C_{2n}^{n} случаев даже на компьютере возможен только для небольших значений n (поэтому последний столбец таблицы заполнен не до конца). Но простыми правилами можно пользоваться и при больших n (и в этом их ценность).

В таблице 1 приведены результаты численных экспериментов по нахождению числа обусловленности для разных способов деления спектра на две части. Из таблицы 1 видно, что наилучшим из простых правил оказывается последнее (выделение в одну группу собственных значений, близких друг к другу). Примеры деления собственных значений на две группы по разным правилам показаны на рис. 2-5.

Компьютерные эксперименты выполнялись в пакете «Математика» [3, 8].

Таблица 1. Значение чисел обусловленности для матриц разного размера и разных способов построения правого корня

Размер матрицы	Первые n значений	Сортировка по возрастанию действительных частей	Сортировка по возрастанию мнимых частей	Группировка по расстоянию	Оптимальный вариант
$n=2$	38.6648	38.6648	1.32376	1.32376	1.10403
$n=4$	44.6692	38.995	174.836	37.3468	1.82899
$n=8$	45072.3	167.184	10.5737	4.91891	4.88812
$n=16$	25182.9	6459.67	1654.99	16.639	
$n=32$	3771.07	155145.3652	620.234	544.43	

Тестовый пример:

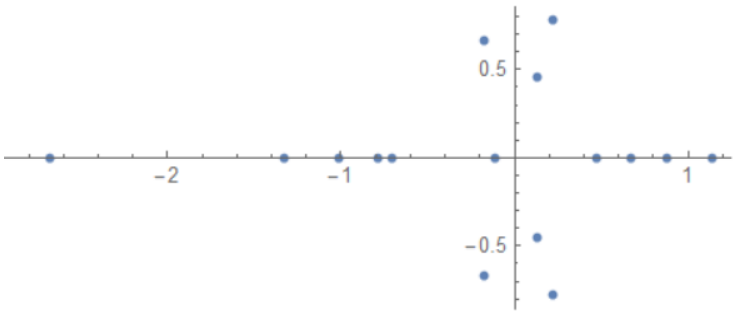


Рис. 10. Собственные значения

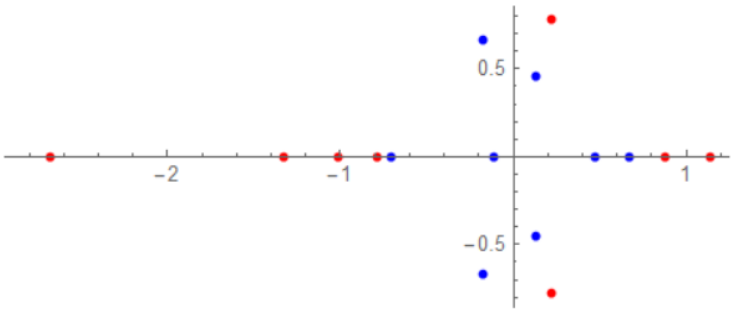


Рис. 11. Первые n значений

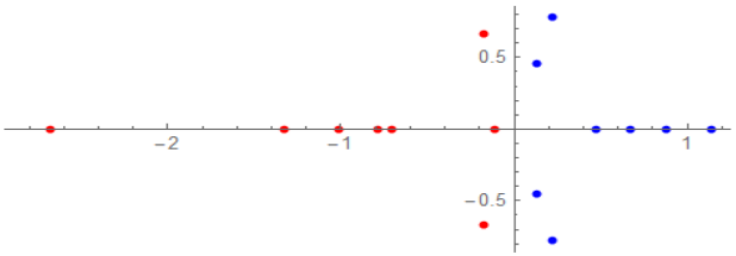


Рис. 12. Сортировка по возрастанию действительных частей

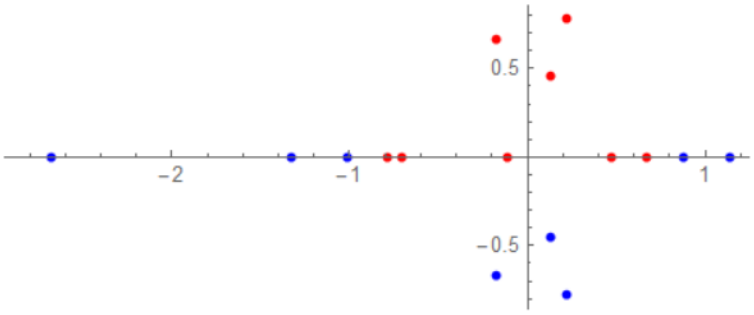


Рис. 13. Сортировка по возрастанию мнимых частей

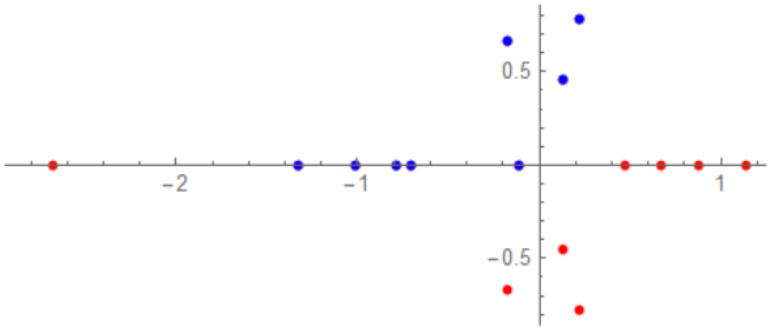


Рис. 14. Группировка по расстоянию

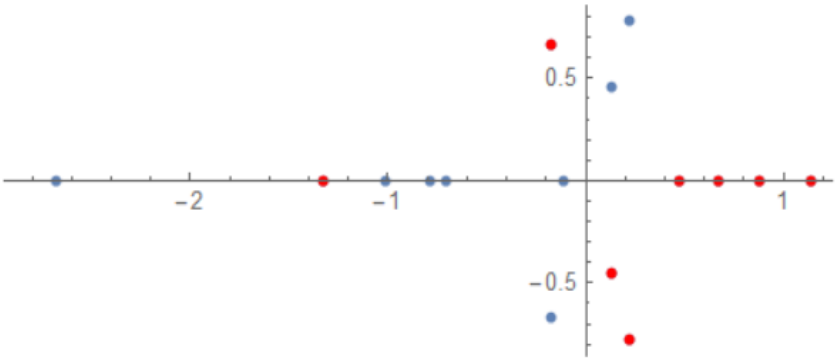


Рис. 15. Оптимальный вариант

Заключение

Проделанная работа позволяет сделать вывод о том, что наиболее эффективным способом для вычисления правого корня матричного пучка является группировка собственных значений и векторов по расстоянию. Числа обусловленности, полученные для этого метода, наиболее близки к оптимально возможным.

Работа выполнена под руководством В.Г. Курбатова, профессора кафедры САиУ Воронежского госуниверситета.

Литература

1. Курбатов, В. Г. Вычислительные методы спектральной теории : учебное пособие / В. Г. Курбатов, И. В. Курбатова. — Воронеж : Издательский дом ВГУ, 2019. — 323 с.
2. Курбатов, В. Г. О нахождении приближенного решения линейного дифференциального уравнения второго порядка / В. Г. Курбатов, М. Н. Орешина // Вестник Воронеж. гос. ун-та. Серия: Физ.-мат. науки. — 2003. — № 2. — С. 173-188.
3. Курбатов, В. Г. Пакет "Математика" в прикладных научных исследованиях / учебное пособие / В. Г. Курбатов, В. Е. Чернов. — Воронеж : Издательский дом ВГУ, 2016. — 240 с.
4. Курбатова, И. В. Функциональное исчисление, порожденное квадратичным операторным пучком / И. В. Курбатова // Записки научных семинаров ПОМИ. — 2011. — Т. 389. — С. 113-130.
5. Перов, А. И. О дифференциальных уравнениях в банаховых алгебрах / учебное пособие / А. И. Перов, И. Д. Коструб. — Воронеж : Издательский дом ВГУ, 2020. — 56 с.
6. Gohberg, I. Matrix polynomials / I. Gohberg, P. Lancaster, L. Rodman. — New York-London : Academic Press, Inc., 1982. — xiv+409 p.
7. Higham, N. J. Numerical analysis of a quadratic matrix equation / N. J. Higham, H. M. Kim // IMA J. Numer. Anal. — 2000. — Vol. 20, no. 4. — P. 499-519.
8. Wolfram, S. The Mathematica book / S. Wolfram. — Fifth edition. — New York : Wolfram Media, 2003. — 1488 p.

УДК 004

СИСТЕМА АВТОМАТИЧЕСКОЙ ДИАГНОСТИКИ ПРЕДСТАРТОВОГО СОСТОЯНИЯ СПОРТСМЕНА «СТРЕССКОНТРОЛЬ»

Д. В. Мозуль, А. А. Гришина, У. И. Усачева

Муниципальное автономное общеобразовательное учреждение «Лицей 44» г.Липецка
Муниципальное бюджетное общеобразовательное учреждение гимназия № 12 г. Липецка
Муниципальное автономное общеобразовательное учреждение «Лицей 44» г.Липецка

Введение

Успех любого спортсмена на соревнованиях зависит от множества факторов, одним из которых является его состояние перед турниром. Известно много примеров, когда, несмотря на хорошие результаты на тренировках, спортсмен оказывался неспособным показать высокий результат. Стресс у спортсменов перед соревнованиями проявляется по-разному. У кого-то он проявляется в виде боевой готовности — состоянии, наиболее благоприятно влияющем на результат. Но, зачастую он проявляется в виде предстартовой лихорадки или предстартовой апатии — состояниях, которые могут пагубно повлиять на результат.

Цель работы: Разработка аппаратно-программного комплекса (АПК) для диагностики предстартового состояния спортсмена с аналитикой стресса и выводом персональных рекомендаций.

Гипотеза: Устройство, определяющее стресс и предлагающее рекомендации по избавлению от состояний предстартовой апатии или лихорадки, поможет привести спортсмена в состояние боевой готовности и настроить на лучший результат.

Практическая значимость: Использование данного АПК будет полезно спортсменам для прогнозирования и оценки оптимального боевого состояния (ОБС). Аппаратные средства АПК рассчитаны на считывание различных показателей, влияющих на стресс и оценку уровня стресса пользователей 14–18 лет.

Актуальность работы: Система автоматической диагностики предстартового состояния спортсмена совсем недавно стали активно использоваться при участии в различных тестированиях, а также данная тема является малоизученной.

1. Теоретические исследования

Предстартовые и стартовые реакции обусловлены эмоциями. Наиболее ярко они проявляются перед крупными соревнованиями. Выделяют 3 предстартовых состояния: стартовая апатия, боевая готовность, стартовая лихорадка.

Стартовая апатия — торможение, которое наступает после чрезмерного возбуждения. Это обратная сторона лихорадки, которая характеризуется неуверенностью спортсмена в своих силах и победе, появлением желания не выходить на старт.

Боевая готовность — идеальная форма стартового состояния. Физиологические сдвиги положительно влияют на интенсивность работы, психологически это проявляется в уверенном ожидании старта, в стремлении одержать победу.

Стартовая лихорадка проявляется в слишком сильном возбуждении нервной системы. Физиологические изменения в этом случае очень велики по сравнению с предстоящей работой. Начинают дрожать руки; случается, что дрожь охватывает все тело.

2. Назначение и функциональные возможности АПК

АПК предназначен для прогнозирования и оценки оптимального боевого состояния (ОБС) у юных спортсменов (14–18 лет). Аппаратные средства АПК рассчитаны на считывание различных показателей, влияющих на стресс и оценку уровня стресса. состоящая из измерительной системы и программного обеспечения для автоматического анализа полученных данных и визуализации результатов.

К основным функциям АПК относятся:

1. Считывание ритмов сердечных сокращений (пульс) и сатурации.
2. Передача данных, полученных в аппаратной части, на сервер в базу данных.
3. Оценка полученных данных — определение типа предстартового состояния спортсмена в зависимости от уровня стресса. (боевая готовность, предстартовая лихорадка или предстартовая лихорадка)
4. Показ полученного результата и вывод рекомендаций по дальнейшим действиям в зависимости от уровня стресса.

3. Аппаратная часть

3.1. Общие сведения об аппаратной части

Аппаратная часть предназначена для снятия и проверки показателей биологических датчиков, определение уровня стресса, оцифровки и отправки данных по Wi-fi на сервер.

Устройство представляет собой: пластиковый корпус прямоугольной формы белого цвета с защелкивающийся крышкой (размер основания: 6 см х 4,3 см, высота: 2,6 см) и зажимом для считывания более точных показаний, расположенным в верхней его части. За включение/выключение отвечает тумблер на верхней стороне корпуса (рис. 1, рис. 2, рис. 3). За питание отвечает 2 батарейки типа «ch2032» скрытые в корпусе. За выполнение всех необходимых задач и подключение модулей отвечают микроконтроллер WeMos D1 mini R1 esp8266, размещенных внутри корпуса (рис. 4).

Рис. 1. Внешний вид устройства

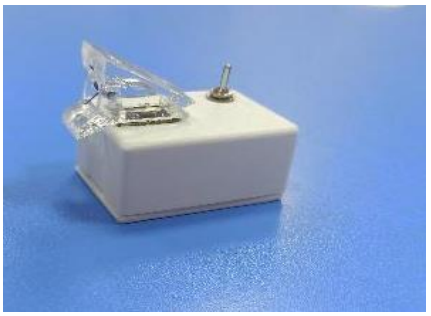


Рис. 2. Внешний вид устройства

Рис. 3. Использование устройства

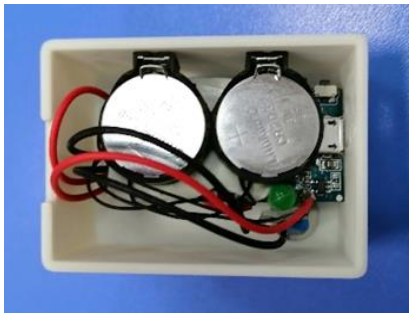


Рис. 4. Внутренний вид устройства

3.2. Функциональное назначение

Устройство, используя цифровой датчик пульса и уровня Sp02 MAX30102 считывает биологические показатели (сатурация, пульс) и сверяет текущие показатели стресса с нормой у определенного спортсмена, после отправляет показатели на сервер для дальнейшей обработки на сайте.

3.3. Используемые детали и модули

Таблица 1

Использованные электронные компоненты

Наименование устройства	Обозначения в схемах	Кол-во
Микроконтроллер WeMos D1 mini R1	wm	1
Цифровой датчик пульса и уровня SpO2, MAX30102, I2C	m	1
Тумблер	tb	1
Батарейный слот типа “ch2032”	bat	2
Провод	pr	8

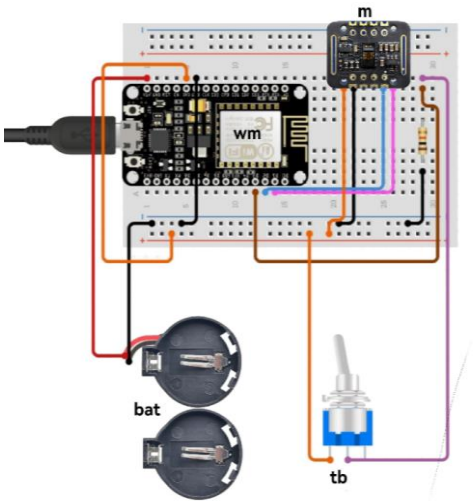


Рис. 6. Функциональная схема

Рис. 7. Логическая схема аппаратной части

4. Программная часть

4.1. Общие сведения о программном обеспечении для аппаратной части

Для обеспечения работоспособности аппаратной части была написана программа на упрощенной версии языка C++ для программирования Arduino. Код устройства реализует следующее: инициализацию библиотек для работы с модулями, создание переменных для записи биологических показателей с датчиков, «setup» с указаниями подключений к портам, обработка биологических показателей (пульса и сатурации), в том числе определение уровня стресса: 1 — боевая готовность, 2 — предстартовая лихорадка, 3 — предстартовая апатия; передача данных на сервер.

Ресурсы программной части: PHP 7.4, CSS 3.0, MYSQL 5.6, HTML 5.3, JS1.8.5.

Веб-приложение разработано на языке гипертекстовой разметки HTML (основная разметка сайта, верстка), языке таблиц стилей CSS (внешний вид и стиль сайта, верстка), а также языках программирования JavaScript (осуществление функций в Frontend) и PHP (осуществление функций в Backend). База данных реализована с использованием системы управления базами данных MySQL.

4.2. Структура веб-приложения с функциональным описанием

Главная страница. Заходя на сайт, пользователь попадает на главную страницу. Если пользователь впервые на сайте, то ему необходимо пройти регистрацию, нажимая на кнопку «Регистрация», в обратном случае, надо нажать на кнопку «Вход» (рис. 8). Также на странице присутствует инструкция по эксплуатации устройства.

Страницы регистрации/входа. Зарегистрированный пользователь по логину и паролю заходит в личный кабинет. При регистрации данные пользователя попадают в базу данных (рис. 9).

Личный кабинет тренера. В личном кабинете тренера отображается: ФИО, тип аккаунта (тренерский/аккаунт спортсмена), список привязанных учеников, окно для привязки спортсменов. Тренер может: привязать аккаунты спортсменов по логину, просмотреть результаты тестирования привязанных спортсменов, нажав на имя спортсмена из списка, выйти из аккаунта (рис. 10).

Личный кабинет спортсмена. В личном кабинете спортсмена изначально отображается: ФИО, тип аккаунта (тренерский, аккаунт спортсмена), ФИО привязанного тренера, окно для привязки устройства. До привязки устройства спортсмен может: привязать устройство, выйти из аккаунта (рис. 11). После привязки устройства спортсмен может: начать тестирование, просмотреть результаты тестирований, выйти из аккаунта (рис. 12, рис. 13).

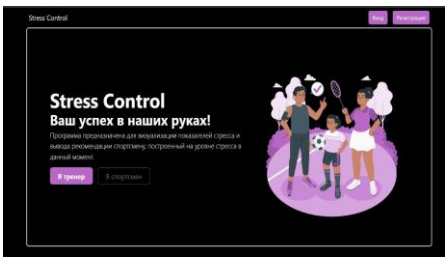


Рис. 8. Главная страница

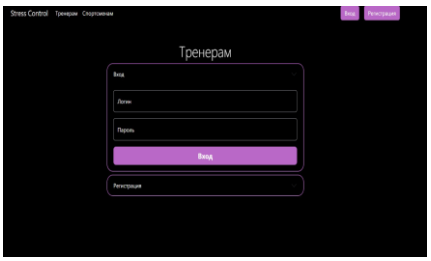


Рис. 9. Страница регистрации

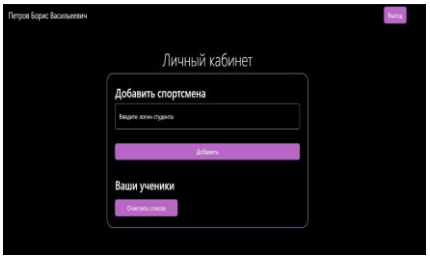


Рис. 10. Личный кабинет тренера

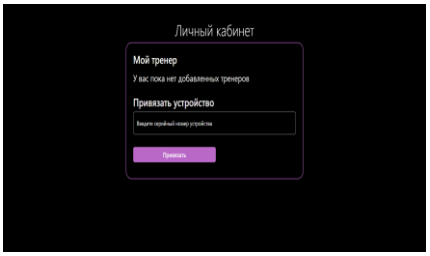


Рис. 11. Личный кабинет спортсмена

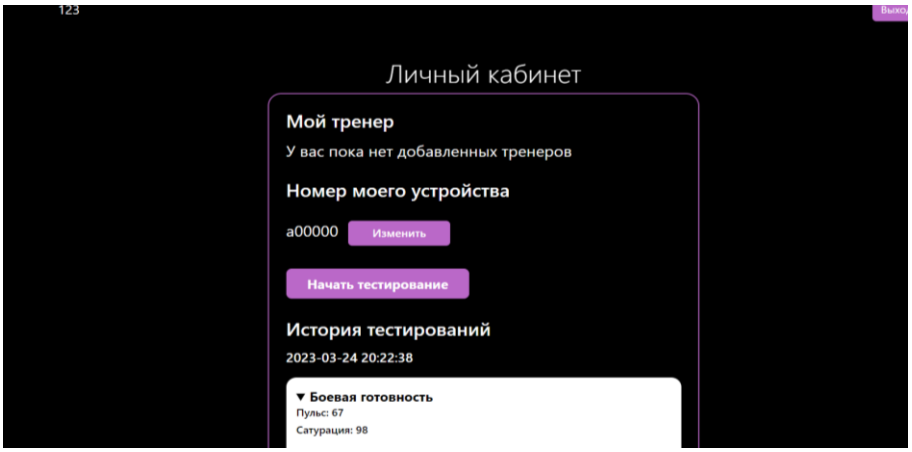


Рис. 12. Интерфейс приложения. Вывод рекомендаций

Рис. 13. Логическая схема программной части

Требования, предъявляемые системой: Wi-fi модуль. Доступ к Интернету. Наличие браузера.

Достоинства аппаратной части: компактность, автономность, удобство использования. Недостатки аппаратной части: высокое потребление энергии, необходимость менять батарейку каждые 2 часа беспрерывных измерений.

Достоинства программного обеспечения: веб-приложение кроссплатформенно, т.е. не предъявляет особых требований для своей работы к аппаратной и программной части, например, операционной системе и браузеру; веб-приложение адаптивно, т.е. корректно отображается на любых устройствах: компьютере, планшете.

Преимущества системы: быстродействие.

Заключение

Аппаратно-программный комплекс предназначен для измерения боевой готовности у юных спортсменов (14–18 лет) перед стартом. Основные функции АПК: считывание параметров физиологических показателей: сатурации и пульса; анализ показателей и вывод

результатов. АПК позволяет определить одно из трех предстартовых состояний: боевая визуализации показателей стресса и вывода рекомендаций. Для взаимодействия пользователя и устройства было разработано веб-приложение, которое доступно по ссылке: stress-kontrol.ru. Благодаря адаптивному дизайну интерфейс сайта корректно отображается на любых устройствах, в том числе на мобильных. На сервере реализована база данных для хранения информации о пользователях, которая обновляется с началом каждой новой сессии.

Список литературы

1. Полиграф. Википедия. – Режим доступа: https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%B3%D1%80%D0%B0%D1%84 – (Дата обращения: 9.01.2023)
2. HUAWEI WATCH fit. – Режим доступа: https://consumer.huawei.com/ru/wearables/watch-fit-new/ – (Дата обращения: 5.01.2023)
3. Apple Watch Series 7. – Режим доступа: https://www.apple.com/ru/apple-watch-series-7 – (Дата обращения: 9.01.2023)
4. Создан карманный измеритель стресса. – Режим доступа: https://www.rbc.ru/society/07/12/2005/5703be089a7947afa08cc46d – (Дата обращения: 5.01.2023)
5. Белов А. В. «Управление модулем Arduino по Wi-Fi с мобильных устройств». — СПб.: Наука и техника, 2020
6. Ханнанов, А.М. Справочник по психологии стресса. – М.: Издательский дом "Эксмо", 2016.
7. Абдуллаев, И.А. Справочник по управлению стрессом. – М.: Омега-Л, 2017.
8. Петрова, Л.А. Стресс и его профилактика. – Москва.: Академия естествознания, 2007.
9. Шалыто, А.А. Стресс. Как с ним бороться. – Москва.: Эксмо, 2016.
10. Макаренко, Е.С. Как пережить стресс: психологические аспекты. - М.: Олма-Пресс, 2011.

УДК 004.89

РАСПОЗНАВАНИЕ АРХИТЕКТУРНЫХ ОБЪЕКТОВ АНТИЧНОЙ ЭПОХИ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ ГЛУБОКОГО ОБУЧЕНИЯ

Ю. В. Молод

Воронежский государственный университет

Введение

Нейронные сети являются объектами пристального внимания пользователей глобальной сети и этому есть логичное объяснение. За последние несколько лет развитие искусственного интеллекта привело к появлению большого числа сложных проектов, функционал которых применяется людьми в их повседневной жизни. Одной из областей применения нейросетей является компьютерное зрение. Благодаря проведенным исследованиям в этой области появилась возможность распознавания различных объектов на изображениях. Самое известное применения подобных технологий – распознавание лиц. Целью данного исследования является создание и обучение нейронной сети, способной отличать на фотографиях античную архитектуру от зданий, относящихся к другой стилистике.

1. Подготовка набора данных

1.1. Поиск изображений

В качестве выборки для исследования были выбраны фотографии зданий, относящихся к периоду античной эпохи, а также фотографии случайных зданий различных временных периодов, так как интерес исследования состоит в обучении сети отличать архитектуру античности от прочей, а не только одного определенного стиля. Основными критериями при подборе фото были высокое качество, размер не менее 500×500 пикселей и различимость архитектурных элементов, принадлежащих стилистике. Все изображения в формате jpeg.

Античная архитектура имеет ряд характерных элементов, благодаря которым ее можно отличить от любой другой. Ее наиболее примечательными образцами можно назвать античные храмы, многие из которых сохранились только частично. Например, от храма Зевса Олимпийского остались только основание, колонны и часть антаблемента. Чуть лучше сохранились храмы Эрехтейон и Парфенон (рис. 1).

Рис. 1. Храм Зевса, Эрехтейон, Парфенон

Ввиду ограниченного числа объектов исследования первоначальная выборка целевых изображений состоит из 50 элементов, многие из которых являются фотографиями разных ракурсов одного объекта. В случае неудовлетворительных результатов обучения нейронной сети объем выборки необходимо будет увеличить.

Основными площадками для сбора стали поисковые системы Google, Yandex, а также сайты стоковых изображений.

1.2. Обработка изображений

После получения достаточного количества изображений их необходимо обработать определенным образом. Каждое изображение приводится к размерам 500x500 пикселей. Обработка производится с помощью средств языка Python и библиотеки OpenCV. Поскольку работа в основном будет проводиться с формой изображаемых объектов не учитывая цвет, необходимо отделить контуры объекта. Для этого дополнительно требуется выполнить следующие действия:

1. Удалить шумы
2. Представить изображение в оттенках серого

Для удаления шумов применим медианный фильтр. Медианный фильтр представляет из себя скользящее окно, размер которого как правило 3x3. На вход принимается значение 9 пикселей, на выход подается одно – медиана среди всех полученных значений.

Представить изображение в оттенках серого позволяет встроенная функция библиотеки OpenCV *BRG2GREY*. Данная функция переводит изображение из трехканального цветного в одноканальное следующим образом: для каждого пикселя значения каналов R, G, и B умножаются на определенный вес и суммируются, тем самым получая одно значение, характеризующее яркость.

После выполнения описанных выше действий осуществляется выделение контуров. Для этого в OpenCV существует функция *findContours*, Чтобы применить функцию, необходимо дополнительно перевести изображение к пороговому представлению – в таком представлении все контуры будут выделены белыми пикселями, черные пиксели будут восприниматься как фон и игнорироваться. К подобному результату может привести применение оператора детекции *Canny*, который выделяет все контуры белым на изображении, а также с помощью функции *threshold*, которая непосредственно является функцией порогового преобразования. Результаты применения *Canny* и *threshold* приведены на рис. 2.

Таким образом, в результате обработки изображений сформирован набор данных одинакового размера, состоящий из обработанных изображений, каждое из которых представляет из себя выделенный белый контур здания на черном фоне.

2. Постановка задачи

Распознавание объектов на изображениях методами глубокого обучения является задачей классификации. Задача классификации состоит из следующих этапов:

1. Разделение выборки на обучающую и тестовую, т. к. будет использоваться тип обучение с учителем.
2. Определение структуры нейронной сети.
3. Выбор оптимизатора, функции потерь и метрики, которые будут использоваться в данной модели.
4. Обучение модели на наборе данных.
5. Анализ полученных результатов.
6. Формулирование вывода.

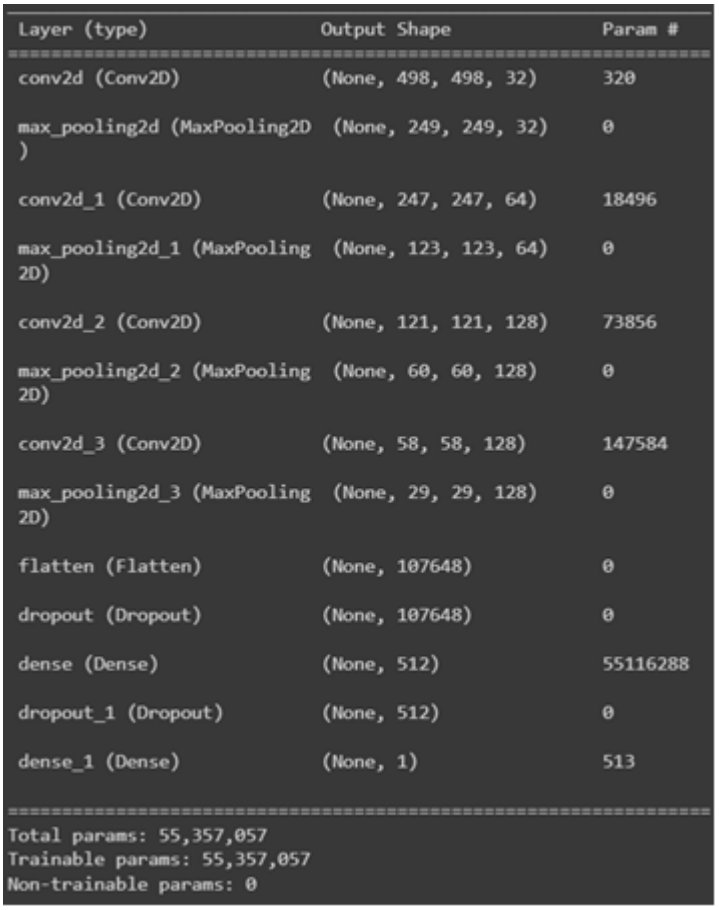
Рис. 2. Сверху вниз результат применения: Canny, threshold

3. Построение нейросети и ее обучение

Построение нейронной сети осуществлялось с помощью библиотеки Keras языка Python в бесплатной интерактивной облачной среде для работы с кодом Google Colab.

Итоговый набор данных состоит из 100 элементов. Из них 50 – изображения античных храмов, оставшиеся 50 – изображения зданий, построенных в другие периоды. Обучающая выборка по объему равна 80 элементам, соответственно тестовая – 20.

Для решения данной задачи использовалась сверточная нейронная сеть. Структура слоев нейронной сети представлена на рис. 3.



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 498, 498, 32)	320
max_pooling2d (MaxPooling2D)	(None, 249, 249, 32)	0
conv2d_1 (Conv2D)	(None, 247, 247, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 123, 123, 64)	0
conv2d_2 (Conv2D)	(None, 121, 121, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 60, 60, 128)	0
conv2d_3 (Conv2D)	(None, 58, 58, 128)	147584
max_pooling2d_3 (MaxPooling2D)	(None, 29, 29, 128)	0
flatten (Flatten)	(None, 107648)	0
dropout (Dropout)	(None, 107648)	0
dense (Dense)	(None, 512)	55116288
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

Total params: 55,357,057
Trainable params: 55,357,057
Non-trainable params: 0

Рис. 3. Структура слоев нейронной сети

Conv2D – слой, осуществляющий свертку. Исходное изображение обрабатывается ядром свертки: ядро сканирует изображение, составляя карту признаков для слоя пикселей. На границе изображение дополняется пикселями с нулевыми значениями так, чтобы ядро не выходило за его пределы (рис. 4). Каждый признак можно описать формулой

$$y_j = \sum_{i=0}^{n} x_i \cdot c_i , \qquad (1)$$

где y_j – значение j-го признака, x_i – значение i-го пикселя из выделенной области, c_i – значение i-го элемента ядра свертки, $i = \overline{1,n}$, $j = \overline{1,m}$, n – количество значений пикселей в

ядре и сканируемой области, m – итоговое количество признаков. Составленная карта признаков будет обработана на следующем слое.

Max_pooling2D - слой, на котором осуществляется операция подвыборки. Данная операция, по сути, является операцией сжатия карты признаков. Для ее осуществления необходимо определить размер выборки и величину шага. С учетом данных параметров карта признаков

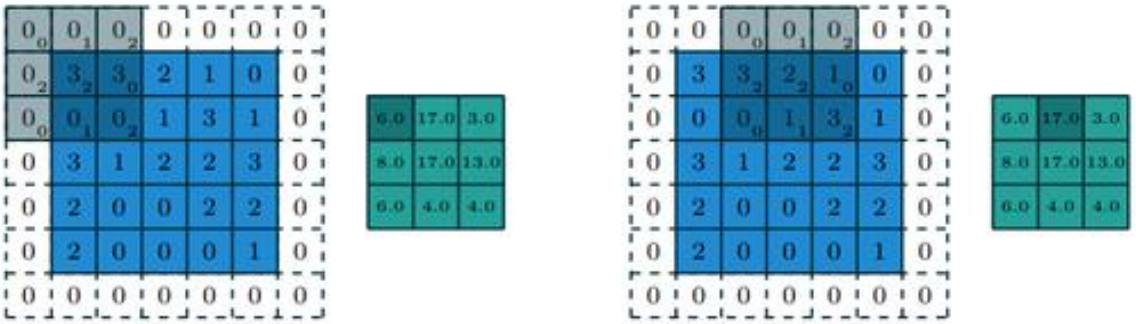


Рис. 4. Получение набора признаков в процессе свертки

будет поделена на выборки, максимальные значения которых составят новую сжатую карту. На рис. 5 для сжатия выбран размер выборки 2x2 и величина шага 2.

Рис. 5. Сжатие карты признаков

Flatten – слой преобразования входных данных из многомерных в одномерные. На данном слое поступающие в виде матриц данные преобразуются в строки. Полученная для каждой матрицы строка состоит из записанных по порядку друг за другом строк исходной формы записи.

Dropout – слой, позволяющий избежать переобучения. Проходящие через него дробные значения признаков заменяются нулевыми.

Dense – полносвязный слой. Выходные нейроны данного слоя связаны с входными по их последнему индексу. Слой выполняет линейное преобразование (2), результатом которого является сумма произведений значений входных нейронов на матрицу весов.

$$y_j = \sum_{i=0}^{n-1} x_i \cdot W_{ij}\,, \qquad (2)$$

где y_j – выходное j-е значение, x_i – значение i-го входного нейрона, W_{ij} – i-й j-й элемент единичной матрицы W, n – размерность последнего индекса.

В качестве оптимизатора выбран алгоритм *Adam*, функции потерь – *categorical_crossentropy*, метрики – *accuracy*.

Adam – один из самых эффективных алгоритмов оптимизации в обучении нейронных сетей. Данный алгоритм вычисляет экспоненциальное скользящее среднее градиента и квадратичный градиент, а его эффективность обусловлена двумя настраиваемыми гиперпараметрами, управляющими скоростью затухания скользящих средних.

Categorical_crossentropy – ошибка на одном примере обучающей выборки. Позволяет оценить качество модели классификации. Значение потери оценивается в диапазоне от 0 до 1, где 0 считается идеальной моделью. Рассчитывается по формуле

$$L(y-y) = -\sum_{i=1}^{K} y_i \cdot \log y_i, \quad (3)$$

где y_i – вероятность принадлежности к классу i, y_i – i-е значение вектора, полученного в результате работы модели, K – количество классов, $i = \overline{1,K}$.

Accuracy – метрика, характеризующая качество модели. Данная метрика описывает общую точность предсказания модели по всем классам. Она рассчитывается как отношение количества правильных прогнозов к их общему количеству.

Построенная модель сверточной нейронной сети была обучена на трех наборах, имеющих описанную выше структуру. Результаты обучения построенной модели приведены в табл. 1.

Таблица 1

Результаты обучения нейронной сети

Набор данных	Accuracy на тестовой выборке
Исходный набор изображений	0.833
Набор изображений, обработанный с помощью Canny	0.916
Набор изображений, обработанный с помощью threshold	0.872

Заключение

В ходе выполнения данной работы была построена и обучена модель глубокого обучения, распознающая на изображениях античные здания. Обучение осуществлялось на 3-х наборах данных, отличных друг от друга методами предварительной обработки. Все наборы показали достаточно высокие результаты, лучшим оказался набор данных, контуры объектов на изображениях которого были получены с помощью оператора Canny.

Данную модель можно использовать в дальнейших исследованиях, адаптировав под распознавание зданий различных архитектурных стилей. Также на ее основе можно построить более совершенные модели, усложнив их применением существующих специальных алгоритмов распознавания или созданным на их основе новым алгоритмом.

Научный руководитель: канд. физ.-мат. наук, ст. преподаватель кафедры Математических методов исследования операций факультета Прикладной математики, информатики и механики Воронежского государственного университета, Замятин Игорь Викторович.

Литература

1. Коул, А. Искусственный интеллект и компьютерное зрение. Реальные проекты на Python, Keras и TensorFlow / А. Коул, С. Ганджу, М. Казам – Санкт-Петербург : Питер, 2023. – 624 с.

2. Келлер А. Изучаем OpenCV 3. / А. Келлер, Г. Брэдски – Москва : ДМК Пресс, 2017. – 826 с.

3. Солем Я. Э. Программирование компьютерного зрения на языке Python / Я. Э. Солем – Москва : ДМК Пресс, 2016. – 312 с.

4. Синельникова А. П. Архитектура Древней Греции / А. П. Синельникова / Вестник науки. – 2020. – № 5. – С. 210–226.

УДК *004.4*

ИССЛЕДОВАНИЕ ПОДХОДОВ ПО УВЕЛИЧЕНИЮ ПРОПУСКНЫХ СПОСОБНОСТЕЙ СЕТЕЙ

А. Н. Нестеров

Воронежский государственный университет

Введение

С ростом городов растут потоки, которые обслуживают городских жителей. К таким потокам можно отнести транспортные, электрические, водные и другие. Но не всегда заранее можно предугадать размеры потоков, которые будут обслужены имеющимися в городе сетями.

Изучение загруженности сетей направлено на выявление потребности увеличения пропускной способности. Перед постройкой новой дороги, участка трубопровода или высоковольтной линии электропередач нужно выбрать тот участок, который даст максимальный прирост потока.

В этой связи возникает необходимость разработки инструмента, с помощью которого можно будет производить вычисления, определяющие порядок добавления новых ребер графа, моделирующего сеть, с учетом стоимости их постройки.

1. Анализ задачи

Необходимо разработать приложение, которое позволяло бы пользователю создавать граф, заполнять его данными и находить оптимальное ребро для увеличения потока сети.

За основу математической модели взята задача о максимальном потоке. Она заключается в определении максимально возможного потока через сеть, состоящую из узлов и ребер, где ребра представляют собой линии связи, которыми передаются данные или энергия. Узлы представляют собой точки передачи или приема этих данных или энергии.

Задача изображается в виде графа, где узлы обозначают начальную точку передачи данных или энергии и конечную точку приема. Ребра задаются пропускной способностью, то есть максимальным количеством данных или энергии, которые могут быть переданы через него.

Основная задача заключается в том, чтобы определить, каким образом можно передать максимально возможное количество данных или энергии через эту сеть, сохраняя заданную пропускную способность каждого ребра. При этом нужно учитывать, что общее количество передаваемых данных или энергии не может превышать пропускной способности наиболее узкого места в сети.

Существует несколько способов решения поставленной задачи.

Алгоритм Форда-Фалкерсона является классическим алгоритмом нахождения максимального потока в сети. Он основан на создании дополнительной сети, которая состоит из остаточных пропускных способностей между вершинами в исходной сети. Сложность алгоритма Форда-Фалкерсона зависит от того, как реализуется поиск увеличивающих путей. Если используется поиск в глубину, то сложность алгоритма составляет $O(E*F)$, где E - количество ребер в графе, а F - величина максимального потока. Однако, есть случаи, когда поиск увеличивающего пути может занимать больше времени.

Алгоритм Эдмондса-Карпа является модификацией алгоритма Форда-Фалкерсона. В этом алгоритме используется поиск в ширину в остаточной сети, что позволяет находить кратчайшие увеличивающие пути. Именно кратчайшие увеличивающие пути обеспечивают асимптотически лучшую производительность алгоритма. Сложность алгоритма Эдмондса-Карпа составляет $O(V*E^2)$, где V - количество вершин, а E - количество ребер в графе. Этот алгоритм гарантирует нахождение максимального потока за полиномиальное время при использовании асимптотически оптимального алгоритма поиска в ширину.

Алгоритм Диница является одним из самых эффективных алгоритмов для нахождения максимального потока в сети. Он основан на использовании слоистой сети, которая создается путем разбиения вершин на уровни в зависимости от расстояния до источника. Сложность алгоритма Диница составляет $O(V^2*E)$, где V - количество вершин, а E - количество ребер в графе. Однако, на практике алгоритм Диница работает гораздо быстрее, чем алгоритмы Форда-Фалкерсона и Эдмондса-Карпа, благодаря использованию слоистой сети и многих оптимизаций.

Разрабатываемое приложение должно позволять пользователю выбирать какой алгоритм использовать для вычисления максимального потока, помимо этого пользователю будет предлагаться набор действий, представленный на рис. 1.

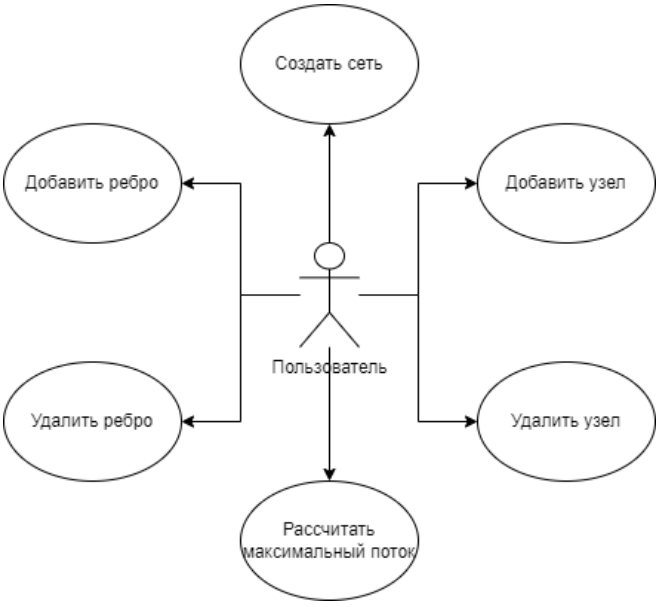


Рис. 1. Диаграмма вариантов использования пользователем

2. Разработка

В качестве языков программирования были выбраны Java и TypeScript. Использовались фреймворки Spring и Angular. Java является кроссплатформенным языком, что позволяет разворачивать приложения в любой среде. Angular является одним из основных фреймворков, использующихся для разработки web-приложений. Одним из преимуществ Angular является наличие библиотек для отображения графов. Для хранения данных была выбрана база данных PostgreSQL. База данных состоит из трех сущностей - сеть, ребро и узел. Сеть содержит основную информацию о сети и права на редактирование и чтение сети. Ребро содержит такие поля как пропускная способности, стоимость данного ребра, тип ребра и оставшаяся пропускная способность. Тип ребра описывает является ли оно основным в сети или новым, для которого будет рассчитываться новая пропускная способность. Узел содержит два поля —

имя и тип. Тип в данном случае говорит о том является ли узел источником, стоком или ни тем и ни другим.

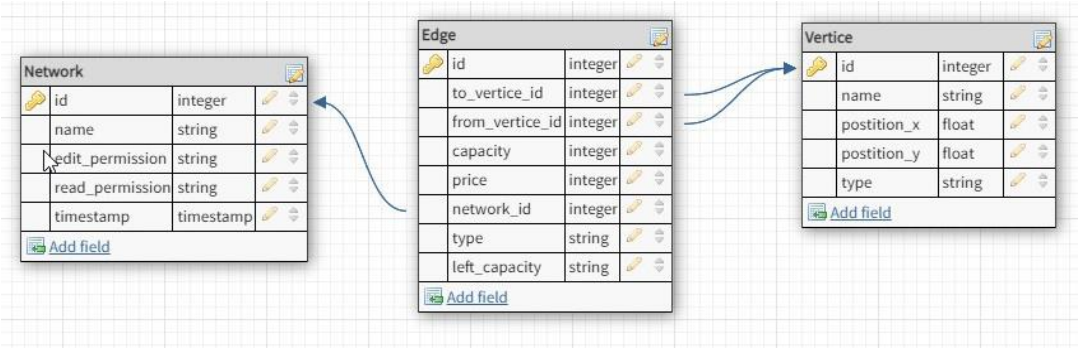


Рис. 2. Схема базы данных

3. Интерфейс приложения

Основной экран приложения состоит из полотна, на котором отображается граф и кнопок, позволяющих редактировать граф.

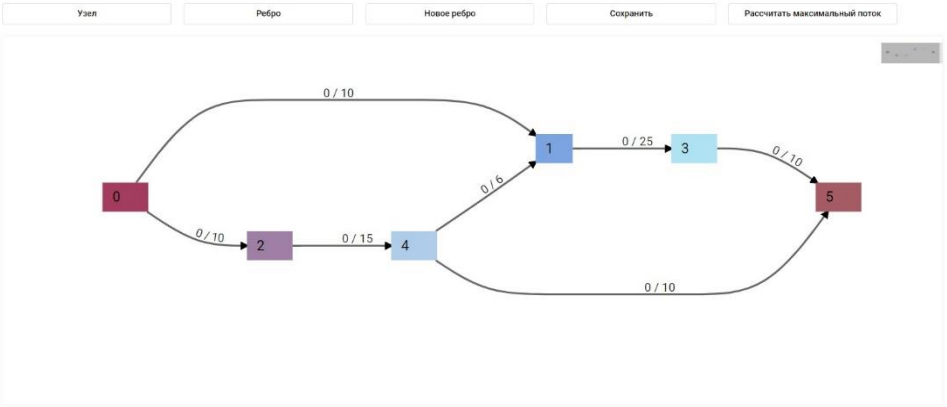


Рис. 3. Построенный граф

После расчета максимального пути ребра графа раскрашиваются в цвета в зависимости от оставшейся пропускной способности ребра. Также на экране выводится статистика с максимальным потоком.

Рис. 4. Граф с рассчитанным максимальным потоком

Заключение

Проанализированы методы поиска максимального потока в сети. Разработанный программный продукт позволяет пользователю задать параметры графа, указать вершины, между которыми планируется построить новое ребро и указать его пропускную способность и стоимость. Пользователю предоставляется возможность выбора алгоритма поиска максимального потока в сети, на основании которого в графе будет выделено оптимальное для постройки ребро и показана новая пропускная способность всей сети.

Литература

1. Липский В. Комбинаторика для программистов: Пер. с польск. – М.: Мир, 1988. – 213 с. ил.
2. Введение в математическое моделирование транспортных потоков: Учебное пособие / Издание 2-е, испр. и доп. А. В. Гасников и др. Под ред. Гасникова. – М.: МЦНМО, 2013 – 362 с.
3. Макоха А. Н., Сахнюк П. А., Червяков Н. И. Дискретная математика: Учеб. пособие. – М., 2005 – 368 с.

УДК 004.023

ИССЛЕДОВАНИЕ МОТИВАЦИИ К ПОЛУЧЕНИЮ ВЫСШЕГО ОБРАЗОВАНИЯ ПРОГРАММНЫМИ СРЕДСТВАМИ АНАЛИЗА КАЧЕСТВЕННЫХ ДАННЫХ НА ОСНОВЕ ДЕТЕРМИНАЦИОННОГО АНАЛИЗА

В. В. Нестругина

Воронежский государственный университет

Введение

В социологических исследованиях для обработки качественных данных применяются различные способы и средства. Задача таких инструментов – изучение структуры связей между переменными, построение вторичных агрегированных показателей [2, 3].

Детерминационный анализ – техника по построению локального, фрагментарного анализа. Он заключается в том, чтобы по одним признакам предсказать наличие других. Главной характеристикой является условная частота. Манипулируя сочетаниями отдельных свойств, можно получить точные и полные детерминации, то есть предсказания [1]. Объектом исследования являются номинальные (качественные) данные, проблема квантификации значений которых в настоящее время не решена.

Предметом исследования служит опросник качества образования для студентов, обучающихся по программам высшего образования, то есть бакалавриата, магистратуры и специалитета, заполненный студентами Воронежского государственного университета.

1. Особенности реализации программных средств для основных задач детерминационного анализа

Задача: методами детерминационного анализа описать зависимости не между переменными (признаками), а между конкретными значениями этих переменных, то есть выяснить, насколько связаны переменные и конкретные значения этих переменных. Для реализации решения были разработаны программные средства, обеспечивающие методами детерминационного анализа обработку данных csv файлов в виде веера отображений (матрицы данных).

Изначально данные для социологического исследования приходят в виде таблицы (матрицы данных), где строки – объекты (E), столбцы – переменные (отображения), пересечение строки и столбца – значение.

Первичные данные исследований представляют собой веер отображений – совокупность отображений вида $E \to X_i, i \in [1, n]$, где $_E$ – множество объектов, X_i – множество значений переменной $_{x_i}$.

Веер образуют компоненты-отображения. Основание веера – множество $_E$. Математические методы анализа социально-экономических данных – методы оперирования веерами отображений. Любое множество $_E$, X_i обладает следующими свойствами: дискретность, конечность [1].

Для анализа первичные данные переводят в таблицы сопряженности (рис. 1)

Оценка условной вероятности или условная эмпирическая частота является наиболее приемлемой мерой установления связи между конкретными признаками и вычисляется следующим образом:

$$P(y \mid x) = \frac{N_{x_i y_j}}{N_{x_i}} \quad (1)$$

В таблице сопряженности каждая клетка рассматривается как изображение прямой и обратной детерминации. ($x_i \to y_j$, $y_j \to x_i$). У каждой детерминации существуют две условные частоты – характеристики этих детерминаций – интенсивность и емкость [1].

Интенсивность детерминации $a \to b$, где $a = x_i$, $b = y_j$ вычисляется по формуле 2:

$$I(x_i \to y_j) = P(y \mid x) = \frac{N_{x_i y_j}}{N_{x_i}} \quad (2)$$

Интенсивность детерминации отражает ее точность/истинность. Она показывает, что среди респондентов, обладающих признаком a, какая-либо доля демонстрируют поведение b [1].

Емкость вычисляется по формуле 3:

$$C(y_j \to x_i) = \frac{N_{x_i y_j}}{N_{y_j}} \quad (3)$$

Емкость показывает, сколько респондентов, или объектов, среди тех, кто демонстрирует тип поведения (признак) b, количество тех, кто является a, $a = x_i$. Иными словами, емкость – это доля реализации поведения b, которая объясняется высказыванием «Из a следует b». Полнота детерминации $a \to b$ – значение емкости, отражающее, насколько всеобъемлюще объяснение, построенное на детерминации $a \to b$.

Метод детерминационного анализа состоит в анализе детерминационных функций. Детерминационная функция образована детерминациями – логическими импликациями (следованиями), порожденными условными частотами. Правило «Если a, то b» – детерминация $a \to b$, то есть b предсказывается на основе a. Детерминация показывает, что одно событие оказывает влияние на другое событие [1].

Основная задача детерминационного анализа – в полном классе D-функций от x к y в контексте k найти все D-функции, которые удовлетворяют ограничениям (5):

$$\begin{cases} I(k(x \xrightarrow{\phi} y)) \geq \delta \\ C(k(x \xrightarrow{\phi} y)) \geq \sigma \end{cases} \quad (4)$$

где δ, σ – некоторые константы, устанавливающие минимальный порог интенсивности и емкости детерминации, причем $0 \leq \delta \leq 1$, $0 \leq \sigma \leq 1$ [1].

Уточнение, которое свойство с вносит в детерминацию $a \to b$ – наличие дополнительного детерминированного свойства c в детерминации $a \to b$. Оно обозначается как $ac \to b$.

Приращение интенсивности – мера существенности уточнения c, вносимого в детерминацию $a \to b$.

Его вычисляют по формуле 5:

$$S(ac^* \to b) = I(ac \to b) - I(a \to b) \quad (5)$$

2. Задача исследования мотивации студентов по получению диплома высшего образования

Система на основе детерминационного анализа была использована для исследования вопроса о том, как изменяется отношение студентов к своему направлению на протяжении всего периода обучения.

В начале была предпринята попытка выяснить, какие курсы наиболее и наименее довольны своим выбором направления обучения. То есть исследовалось отношение студентов к корректности выбора образовательной программы. Был взят универсальный контекст

(рассматривались все студенты университета, вне зависимости от факультета и направления обучения), набор переменных X состоит из одной переменной курса (значения от 1 до 6, где 5 и 6 – 1 и 2 курсы магистратуры, соответственно). Набор переменных для Y состоит из переменной «Если бы вам снова представилась возможность выбирать направление обучения, вы бы:», и последовательно исследуются ее два значения «повторил бы снова свой выбор» и «не стал бы учиться вообще».

Среди респондентов, обладающих наиболее позитивным отношением к выбору своему направлению, наибольшую интенсивность имел первый курс, и чем старше курс бакалавриата, тем больше падает интенсивность детерминации «Если студент обучается на определенном курсе, то он будет чувствовать, что повторил бы выбор своего направления обучения снова» падает с 0.73 до 0.573. Прогресс в обучении по программам также демонстрирует падение точности детерминации. Наибольшей полнотой обладает значение X = первый курс, что говорит о том, что первый курс имеет наибольшую долю реализации поведения «Повторил бы снова свой выбор» среди всех курсов.

Таблица 1

Детерминация «Если бы я был на определенном курсе, то выбрал бы ту же специальность»

Курс	I	C	N(X&Y)	N(X)	N(Y)
1	0.730	0.363	2096	2871	5779
2	0.674	0.234	1353	2006	5779
3	0.632	0.182	1054	1668	5779
4	0.573	0.159	918	1602	5779
5	0.600	0.045	262	437	5779
6	0.560	0.015	89	159	5779

Если в качестве значения взять значение переменной Y, равное «Не стал бы учиться вообще», то, хотя и точность такой детерминации низка (менее 0.05 среди всех курсов), однако можно проследить, какие из значений X составляют наиболее полную детерминацию, и здесь это – первый и второй курс.

Таблица 2

Детерминация «Если бы я был на определенном курсе, то не стал бы учиться вообще»

Курс	I	C	N(X&Y)	N(X)	N(Y)
1	0.025	0.287	72	2871	251
2	0.036	0.291	73	2006	251
3	0.017	0.112	28	1668	251
4	0.031	0.199	50	1602	251
5	0.050	0.088	22	437	251
6	0.038	0.024	6	159	251

При выборе контекста в виде переменной факультета, значения «Факультет прикладной математики, информатики и механики», при Y = «Повторил бы снова свой выбор» можно увидеть, что для первого и второго курса интенсивность меньше, то есть контекст отрицательно уточняет переменную курса, а для остальных курсов – положительно.

Таблица 3

Детерминация «Если бы я был на определенном курсе то выбрал бы ту же специальность» в контексте факультета

Курс	I	C	N(X&Y)	N(X)	N(Y)
1	0.687	0.327	101	147	309

2	0.709	0.291	90	127	309
3	0.506	0.146	45	89	309
4	0.655	0.184	57	87	309
5	0.800	0.026	8	10	309
6	0.615	0.026	8	13	309

При выборе Y = «Не стал бы учиться вообще» контекст для большинства курсов отрицательно уточняет детерминацию «Если человек на определенном курсе, то он считает, что не стал бы учиться вообще», и наиболее негативным отношением к образованию обладают те, кто учится на 2 курсе магистратуры.

Таблица 4

Детерминация «Если бы я был на определенном курсе, то не стал бы учиться вообще» в контексте факультета

Курс	I	C	N(X&Y)	N(X)	N(Y)
1	0.007	0.091	1	147	11
2	0.031	0.364	4	127	11
3	0.022	0.182	2	89	11
4	0.023	0.182	2	87	11
5	0.000	0.000	0	10	11
6	0.154	0.182	2	13	11

Второй задачей являлось получение уточнения для детерминации «Если человек на определенном курсе, то он считает, что не стал бы учиться вообще» на основе переменной «Чем вы руководствуетесь, получая образование? (Можно выбрать несколько вариантов ответов)». Так как это переменная со множественным выбором, то сначала производится переход к набору переменных с единственным выбором, где переменная – вариант ответа на данный вопрос, а значение – 1 или 0 – выбрал ли респондент этот вариант или нет, соответственно. Затем рассматриваются некоторые из вариантов ответа и с помощью задачи определения приращения интенсивности находится, какое приращение интенсивности, то есть уточнение детерминации, даст данный вариант.

Таблица 5

Сравнение приращений интенсивности при разных уточнениях детерминации

Курс	Y = «Повторил бы снова свой выбор»		Y = «Не стал бы учиться вообще»	
	Уточнение «Стремлюсь стать специалистом, востребованным на рынке труда»	Уточнение «Мне нужен диплом, а знания не так уж и важны»	Уточнение «Стремлюсь стать специалистом, востребованным на рынке труда»	Уточнение «Мне нужен диплом, а знания не так уж и важны»
1	0.057	-0.379	-0.005	0.085
4	0.097	-0.242	-0.016	0.086

Для респондентов, которые наиболее удовлетворены выбором своей специальности (Y = «Повторил бы снова свой выбор», наибольшее уточнение (наибольшее положительное приращение интенсивности) вносит уточнение «Стремлюсь стать специалистом, востребованным на рынке труда», тогда как, судя по наибольшему уточнению Y = «Не стал бы учиться вообще» вариантом «Мне нужен диплом, а знания не так уж и важны», респонденты,

которые недовольны своей специальностью, наиболее точно описываются детерминацией «Если студент на 4 курсе и руководствуется тем, что ему просто нужен диплом, а не знания, совершенно недоволен своим выбором специальности, да и самим фактом получения высшего образования.

Заключение

Была проанализирована зависимость между отдельными значениями номинальных переменных в исследовании поведения студентов, обучающихся по программам высшего образования. При анализе были использованы программные средства на основе детерминационного анализа, которые позволяют делать выводы о зависимости между конкретными значениями номинальных переменных, используя для этого условные частоты и их разности, составлять на их основе детерминации и дифференцировать различные свойства по степени существенности их вклада в аргументы детерминации, измеряемой величинами соответствующих приращений условных частот. Вопросы, поставленные для выявления связи между переменными, были решены посредством разработанного приложения, которое поддерживает решение основных задач детерминационного анализа и агрегированных задач на базе основных.

Список литературы

1. Чесноков С. В. Детерминационный анализ / С. В. Чесноков ; под ред. Е. С. Райской. – Москва : Наука. Главная редакция физико-математической литературы, 1982. – 168 с.
2. Бородкин Ф. М. Математическое моделирование в социологии (методы и задачи) / Ф. М. Бородкин, Б. Г. Миркин. – Новосибирск : Наука, 1977. – 240 с.
3. Чесноков С. В. Взаимоотношение теоретического и эмпирического уровней описания социально-экономической реальности / С. В. Чесноков // Методология комплексного исследования социально-экономических систем. Труды ВНИИСИ ГКНТ и АН СССР. – Москва, 1980. – Вып. 1. – С. 33–42.

УДК 004.8

ДЕТЕКЦИЯ ЧЕЛОВЕЧЕСКИХ СИЛУЭТОВ НА СНИМКАХ, ПОЛУЧЕННЫХ С ПОМОЩЬЮ БПЛА.

А. Е. Ни

Воронежский государственный университет

Введение

Детекция человеческих снимков - это процесс автоматической обработки изображений с целью обнаружения на них человеческих объектов или лиц. Этот процесс является важным компонентом во многих приложениях компьютерного зрения, таких как системы видеонаблюдения, робототехника, медицинская диагностика и другие.

В последние годы использование беспилотных летательных аппаратов (БПЛА) стало все более распространенным. Одной из важных областей применения является съемка лесного массива для оценки его состояния и мониторинга растительности. Однако, на снимках леса, полученных с помощью БПЛА, часто трудно определить наличие людей.

Основным методом детекции человека на снимках является использование алгоритмов машинного обучения, в частности, нейронных сетей. Эти алгоритмы позволяют извлекать признаки из изображения и обучаться на наборе примеров, после чего они могут определять наличие или отсутствие человека с высокой точностью.

В целом, эффективность детекции человека на изображении зависит от точности алгоритмов и методов, использованных при обработке изображения. Обычно, для улучшения работы алгоритмов различные методы обработки изображения, такие как сегментация, фильтрация и кластеризация, могут использоваться в сочетании с методами машинного обучения, для достижения наилучшего результата.

Целью статьи является разработка метода, способного отыскать человеческий силуэт на изображениях, полученных с помощью БПЛА, а также указать количество найденных объектов.

1. Исходные данные

На вход программы нам подаются изображения в лесу, на некоторых из них располагаются люди, на некоторых изображениях их нет, пример изображений показан в рисунках 1 и 2. Всего дано 5163 изображений разного размера с БПЛА, из них 58 с изображением людей.

Рис.1. Пример изображения лесного массива с человеком.

Рис. 2. Пример изображения лесного массива без людей.

Все фотографии были поделены на две папки - test и train. В папке test – содержатся фотографии лесного массива для предсказания, а в папке train – фотографии лесного массива для обучения.

Так же мы имеем два файла, формата Excel.

Файл sample_solution.csv — файл, содержащий данные о фотографиях тестового набора.

	ID_img	region_shape
0	1.JPG	0
1	2.JPG	0
2	3.JPG	0
3	4.JPG	0
4	5.JPG	0

Рис.3. Файл sample_solution.csv

Файл train.csv — файл, содержащий данные о количестве людей на изображении.

	ID_img	count_region
0	3376.JPG	0.0
1	3377.JPG	0.0
2	3378.JPG	0.0
3	3379.JPG	0.0
4	3380.JPG	0.0
...	...	...

Рис.4. Файл train.csv

2. Нейронные сети

Начнем с того, что такое нейронные сети. Они возникли из области искусственного интеллекта. Их прототипом послужили настоящие биологические нейронные сети. Это математическая модель, состоящая из некоторого количества простых взаимосвязанных элементов, обрабатывающих информацию путем изменения своих состояний (параметров системы) в ответ на внешние данные. Они существуют для того, чтобы решать различные задачи по распознаванию объектов или прогнозирования событий. С их помощью можно выявлять различные закономерности для статистических задач[1].

Так как данная задача связана с изображениями, больший интерес уделим сверточным нейронным сетям. Сверточная нейронная сеть (CNN) – это архитектура искусственных нейронных сетей, она нацелена на работу с изображениями.

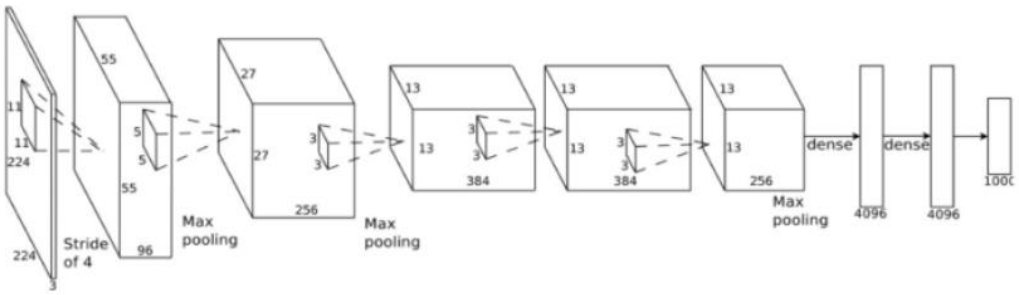


Рис.5. Архитектура сверточной нейронной сети.

Сверточная нейронная сеть состоит из чередующихся между собой сверточных слоев и субдискретизирующих слоев. Первые действуют на входное изображение набором фильтров (ядер), такой процесс называется сверткой. После чего субдискретизирующие слои уменьшают размерность получившегося выхода сверточного слоя. Выходы сверточных и субдискретизирующих слоев называются карты. Карта признаков показывает насколько входное изображение похоже на то или иное ядро.

Таким образом, свертка — это всего лишь линейное преобразование входных данных особого вида. Если x^i — карта признаков в слое под номером l, то результат двумерной свертки с ядром размера $2d+1$ и матрицей весов W размера $(2d+1)*(2d+1)$ на следующем слое будет таким:

$$y^l_{i,j} = \sum_{-d \le a,b \le d} W_{a,b}\, x^l_{i+a,j+b}$$

где — $y^l_{i,j}$ результат свертки на уровне l , а $x^l_{i,j}$ — ее вход, то есть выход всего предыдущего слоя. Иначе говоря, чтобы получить компоненту (i, j) следующего уровня, мы применяем линейное преобразование к окну предыдущего уровня, то есть скалярно умножаем пикселы из окна на вектор свертки. В некоторых моделях CNN после нескольких блоков сверточных слоев могут следовать полносвязные слои, которые представляют из себя многослойный перцептрон. Перцептрон является самой простой вариацией нейронной сети Элементарной частицей перцептрона является нейрон – это элемент, хранящий в себе некоторый вес, умеющий принимать значение на вход, выполнять операцию умножения и выдавать результат на выход. Перцептрон состоит из нескольких слоев, где слой – это набор нейронов, имеющих общий

набор входов и не имеющих связей между собой. На нейроны первого слоя подаются входные данные в виде вектора значений. Далее они умножаются на хранящийся в каждом нейроне вес, а результаты передаются в каждый из элементов следующего слоя. И так далее до последнего слоя, на котором формируется результирующий вектор значений.

3. Загрузка данных в проект.

Для того, чтобы загрузить данные, нам понадобится Google Диск. Google Диск - это облачное хранилище файлов, которое предоставляет Google. С помощью Google Диска пользователи могут загружать, хранить, синхронизировать и обмениваться файлами и папками в Интернете. Google Диск доступен как веб-приложение и как приложение для компьютеров и мобильных устройств. Пользователи могут добавлять файлы к своей учетной записи Google и получать доступ к ним с любого устройства или любой локации с доступом в Интернет[2].
Загрузим все данные, которые у нас есть на Google Диск.

Для дальнейшей работы с программой, нам нужно подключить определенные библиотеки. Рассмотрим главные из них:

1. Pandas - это библиотека для языка программирования Python, предназначенная для обработки и анализа данных.
2. NumPy (Numerical Python) - это библиотека для работы с многомерными массивами и матрицами, которая предоставляет различные математические функции для работы с ними.
3. Keras - это высокоуровневая библиотека глубокого обучения, написанная на языке Python, которая упрощает процесс создания нейронных сетей.
4. Sklearn (Scikit-learn) - это библиотека машинного обучения для Python, которая является открытым исходным кодом и распространяется под лицензией BSD[3].

5. Построение модели

Существует большое количество моделей для обучения. Модели для обучения детекции человека могут содержать следующие элементы[4]:

1. Слои свертки: они извлекают признаки из исходного изображения, позволяя обнаруживать различные особенности человеческой фигуры, такие как форма тела, голова, руки, и т.д.
2. Слои объединения: они уменьшают размерность изображения, что ускоряет процесс обучения и позволяет более эффективно выделять признаки.
3. Слои активации: они добавляют нелинейность в модель, что позволяет ей лучше адаптироваться к сложным сценам, содержащим людей.
4. Слои обнаружения: они используются для выделения окон, которые содержат человеческие фигуры. В зависимости от выбранного алгоритма обнаружения, может использоваться, например, YOLO или Faster R-CNN.
5. Слои классификации: они позволяют модели определить, является ли объект на изображении человеком или нет.
6. Слои регрессии: они используются для определения координат границ объекта на изображении.
7. Слои потерь: они определяют, насколько точно модель делает предсказания, и используются для обучения модели.

В зависимости от конкретной задачи детекции человека, некоторые из вышеперечисленных элементов могут быть изменены или исключены.

В нашем случае, в качестве модели для обучения возьмем модель Keras Sequential. Keras Sequential - это удобный способ определения модели нейронной сети в Keras. Он позволяет последовательно добавлять слои и описывать архитектуру нейронной сети.

```
def get_model():
  model = keras.models.Sequential()

  model.add(Conv2D(32, (3, 3), padding="same",input_shape=[image_size, image_size, 3]))
  model.add(Activation("relu"))
  model.add(BatchNormalization(axis=-1))
  model.add(MaxPooling2D(pool_size=(3, 3)))
  model.add(Dropout(0.25))

  model.add(Conv2D(64, (3, 3), padding="same"))
  model.add(Activation("relu"))
  model.add(BatchNormalization(axis=-1))

  model.add(Conv2D(64, (3, 3), padding="same"))
  model.add(Activation("relu"))
  model.add(BatchNormalization(axis=-1))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))

  model.add(Conv2D(128, (3, 3), padding="same"))
  model.add(Activation("relu"))
  model.add(BatchNormalization(axis=-1))

  model.add(Conv2D(128, (3, 3), padding="same"))
  model.add(Activation("relu"))
  model.add(BatchNormalization(axis=-1))

  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))
  model.add(Flatten())
  model.add(Dense(1024))
  model.add(Activation("relu"))
  model.add(BatchNormalization())
  model.add(Dropout(0.5))

  model.add(Dense(9))
  model.add(Activation("sigmoid"))

  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=
                [keras.metrics.AUC(name='roc_auc', curve = 'ROC')])

  return model
```

Рис.6. Модель в нашей программе.

Данная модель является сверточной нейронной сетью для многоклассовой классификации изображений. Она состоит из нескольких слоев, включая сверточные, пуллинговые, dropout, плотные и др. С помощью функции активации relu и нормализации batch normalization, модель обучается распознавать особенности изображений и классифицировать их в соответствующие классы. Также, задана функция потерь "binary_crossentropy" и

оптимизатор "adam" для обучения модели. Метрика ROC_AUC используется для оценки качества модели.

После обучения модели мы получили точность равную 0.8679 результат представлен на рисунке 7.

```
Epoch 1/8
22/22 [==============================] - 217s 10s/step - loss: 0.9506 - roc_auc: 0.5482
Epoch 2/8
22/22 [==============================] - 207s 9s/step - loss: 0.7127 - roc_auc: 0.6181
Epoch 3/8
22/22 [==============================] - 202s 9s/step - loss: 0.5797 - roc_auc: 0.7110
Epoch 4/8
22/22 [==============================] - 203s 9s/step - loss: 0.4790 - roc_auc: 0.7844
Epoch 5/8
22/22 [==============================] - 203s 9s/step - loss: 0.4261 - roc_auc: 0.8254
Epoch 6/8
22/22 [==============================] - 202s 9s/step - loss: 0.4004 - roc_auc: 0.8432
Epoch 7/8
22/22 [==============================] - 205s 9s/step - loss: 0.3831 - roc_auc: 0.8606
Epoch 8/8
22/22 [==============================] - 208s 9s/step - loss: 0.3738 - roc_auc: 0.8679
CPU times: user 38min 34s, sys: 4min 14s, total: 42min 49s
Wall time: 32min 8s
```

Рис.7. Результат обучения модели.

После пройденных операций, все данные полученные в программе, должны быть выгружены в отдельный файл. В нашем случае это будет файл формата Excel. Будет выполнено прогнозирование вероятности отнесения каждого объекта тестовой выборки к каждому из классов.

После прогнозирования происходит перевод интерпретации вероятности в классификацию с использованием порога 0,5. Затем полученный предсказанный класс конвертируется в формат pandas DataFrame и сохраняется в файл y_preds.csv в Google Drive.

Заключение

В ходе данного исследования мы рассмотрели исходные данные, проанализировали их, а также, построили модель, способную на выходе своей работы выдать нам файл с изображениями, поделенные на те, в которых есть человек и в котором нет с точностью 0.8679.

Стоит отметить, что построение модели детекции человека на снимках является важной задачей в области компьютерного зрения, которая находит широкое применение в различных сферах, включая безопасность, медицину и развлечения. В процессе создания модели необходимы глубокие знания в области машинного обучения и нейронных сетей. В будущем, с развитием технологий, ожидается появление новых методов детекции человека, которые будут еще более точными и удобными в использовании. Однако, необходимо учитывать потенциальные риски и вызовы в области приватности и безопасности, связанные с использованием таких технологий. Поэтому, для обеспечения эффективного и ответственного использования методов детекции человека, важно разработать соответствующие правила и механизмы контроля.

Литература

1. Осовский Станислав. Нейронные сети для обработки информации = Sieci neuronowe do przetwarzania informacji (польск.) / Перевод И. Д. Рудинского. — М.: Финансы и статистика, 2004. — 344 с.
2. Google Disk. — Режим доступа: https://ru.wikipedia.org/wiki/Google_Диск
3. Орельен Жерон. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем = Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems. — Вильямс, 2018. — 688 с.
4. Создание модели машинного обучения. — Режим доступа: https://medium.com/nuances-of-programming/создание-модели-машинного-обучения-с-помощью-google-colab-без-дополнительных-настроек-f5f68f3bd4ac

УДК 004.91

РАЗРАБОТКА МОДУЛЯ ЭЛЕКТРОННОЙ ОТЧЕТНОСТИ ОРГАНИЗАЦИИ

П. А. Никольников

Воронежский государственный университет

Введение

Требования в разных организациях к ведению документации отличается. Каждая фирма хочет получать в результате необходимую для себя информацию, исключая ненужные данные. Большинство существующих программных решений не соответствуют требованиям компаний или нуждаются в серьезной доработке. Также часто возникает проблема интеграции новой функциональности в используемую систему.

1. Описание системы

В рамках электронного документооборота может потребоваться различный набор функций. Так, в организации возникла следующая задача: создать и внедрить в используемую систему Oracle E-Business Suite встраиваемый модуль для формирования авансовой отчетности.

Oracle E-Business Suite (OEBS) — пакет приложений электронного бизнеса, тиражируемый интегрированный комплекс прикладного программного обеспечения производства компании Oracle, включающий функциональные блоки планирования ресурсов предприятия (ERP), управления взаимоотношениями с клиентами (CRM), управление жизненным циклом продукта (PLM)[1]. Предназначен для автоматизации основных направлений деятельности предприятий, в том числе: финансов, производства, управления персоналом, логистики, маркетинга, сбыта и продаж, обслуживания заказчиков, взаимоотношений с поставщиками и клиентами.

Oracle E-Business Suite R12 — архитектура, представляющая структуру для многоуровневых, распределенных вычислений. Состоит из трех уровней вычислений: уровень базы данных (БД), уровень приложений, уровень клиента. Клиентский уровень представлен в виде Java-плагина для веб-браузера, обеспечивающий интерфейс пользователя, мобильного устройства, веб-браузера[2].

Пользователи входят в систему с домашней страницы, через веб-браузер. Система Oracle E-Business Suite R12 обеспечивает единую точку входа во все приложения на основе HTML и на основе Oracle Forms.

Клиентский апплет для Forms представляет собой набор JAR файлов. JAR файлы содержат все необходимые Java-классы для запуска приложений на основе Oracle Forms. Клиентский апплет и наиболее часто используемые JAR файлы загружаются с веб-сервера при запуске Oracle Forms приложений. Другие, менее используемые файлы, загружаются при необходимости.

2. Исходный бизнес-процесс

Изначально пользователь заходил в Oracle Forms. Открывалась форма, на которой пользователь заполнял поля нескольких таблиц и запускал генерацию: «Печатная форма Авансового отчета» и «Заявление на удержание из заработной платы». После этого он выгружал шаблоны необходимых отчетов, это мог быть «Детальный отчет о командировке» или «Отчет о проведении представительских расходов». Пользователь заполнял эти шаблоны данными и относил пакет документов на подписание определенным лицам, перечень которых необходим для каждого отчета (рис. 1, 2).

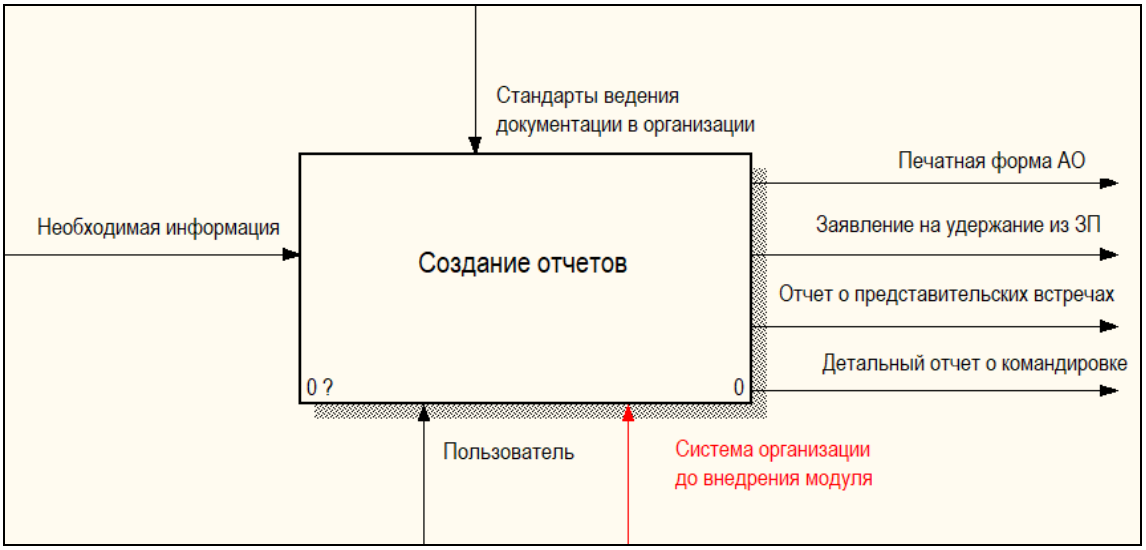


Рис. 1. Исходный бизнес-процесс (общий вид)

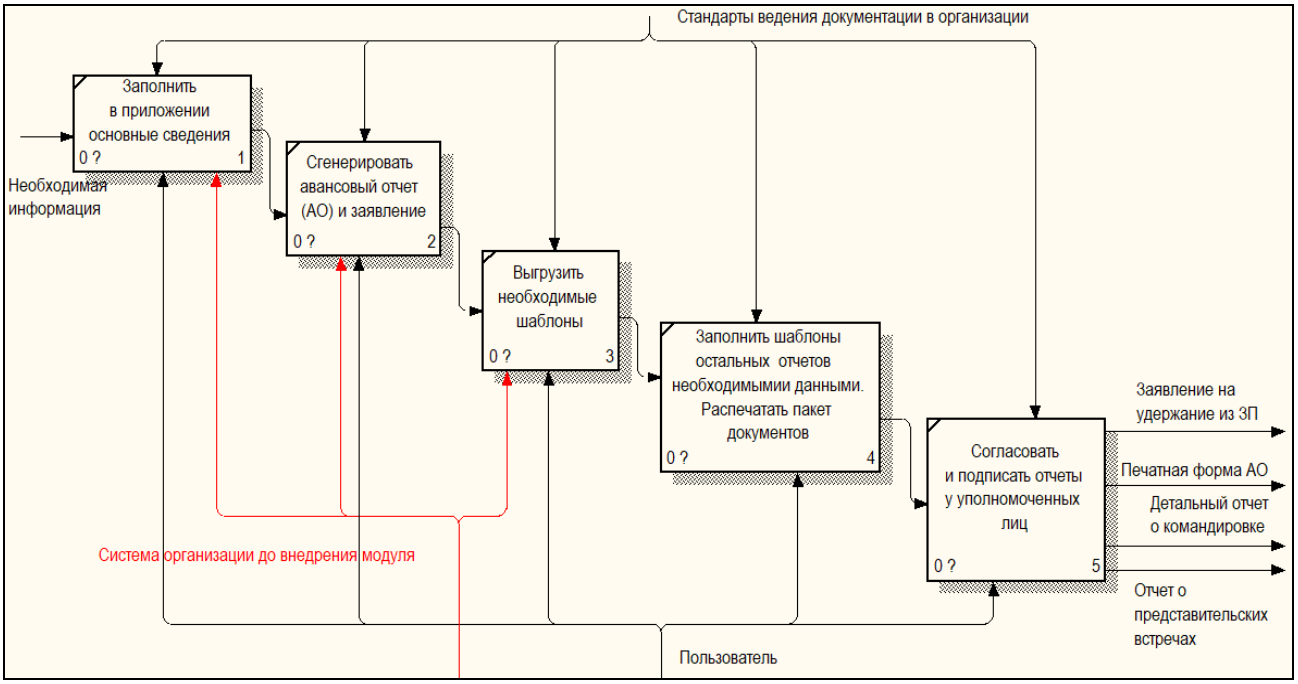


Рис. 2. Исходный бизнес-процесс

В существовавшем процессе (рис. 2) было выявлено множество недостатков. Частичная автоматизация процесса вынуждала пользователя заполнять документы вручную. После приходилось лично ходить к каждому человеку из списка лиц, которые должны согласовать и утвердить документ. Эти уполномоченные лица могли находиться в разных офисах, или отклонить отчет. В этом случае пользователь устранял недочеты и возвращался обратно.

После, некоторые отчеты следовало отнести в бухгалтерию, для дальнейшей работы с ними. Весь процесс занимал у пользователя много времени. Создание отчета в Oracle Forms требовало сбора информации из разных источников, также необходимо было знать об их местоположении и порядке доступа к источникам.

3. Переработанный бизнес-процесс

Следует заметить, что система поддерживала возможность подписывать формы документов в электронном формате. Было решено оптимизировать процесс формирования отчетов. Для этого был разработан специальный встраиваемый модуль. Для его реализации использовались следующие средства разработки. PL/SQL Developer — интегрированная среда разработки для СУБД Oracle. Oracle JDeveloper — бесплатная интегрированная среда разработки программного обеспечения, предоставляющая возможность для разработки на языках программирования Java, JavaScript, BPEL, PHP, SQL, PL/SQL и на языках разметки HTML, XML. Notepad++ — свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса и разметки.

Новый процесс формирования и подписания отчетности реализован через веб-расширение Oracle Application Framework (OAF)[3]. Работник заходит на OAF, в специальном разделе заполняет пользовательскую форму, где указывает все необходимые данные. На основе этой формы создается вторая форма с необходимой для бухгалтера информацией. Запускается процесс электронного утверждения, уже реализованный в другом модуле системы. После согласования форма утверждается электронной подписью. Если все определенные лица из советующего списка поставили электронную подпись, то пользователь может распечатать все нужные документы, которые соответствуют требованиям организации к ведению отчетности. В случаях, когда электронное подписание отменено одним из ответственных лиц, пользователь печатает пакет документов со всеми заполненными данными и идет собирать подписи у уполномоченных руководителей (рис. 3).

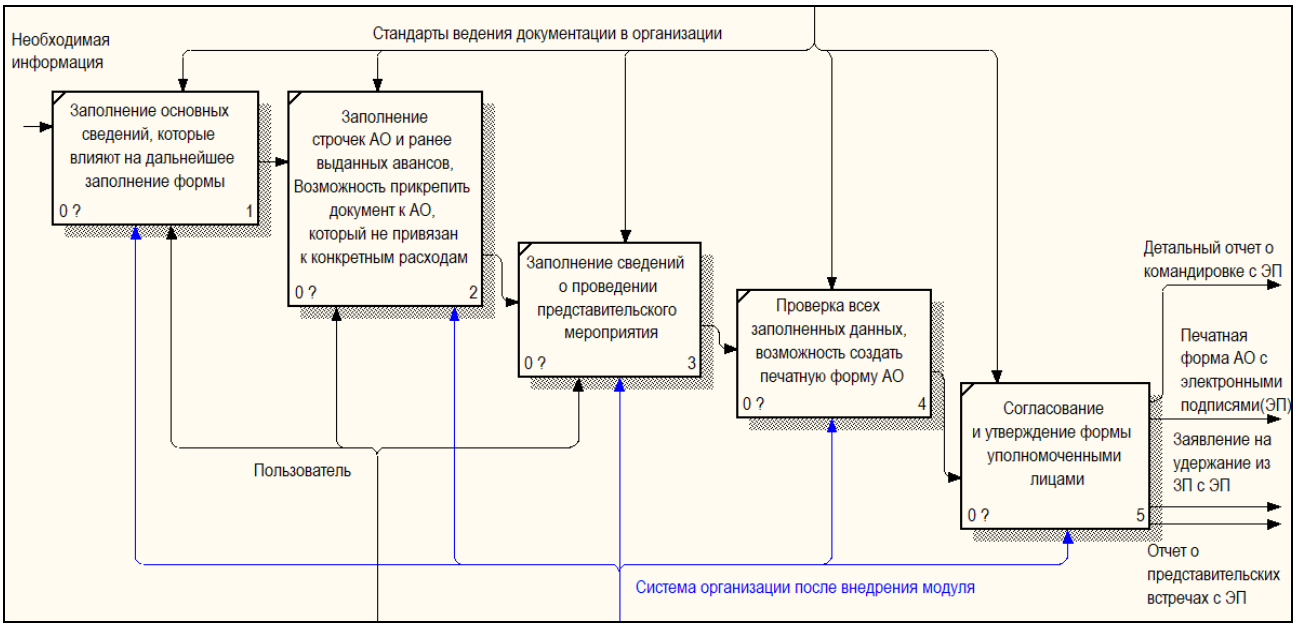


Рис. 3. Переработанный бизнес-процесс

Заключение

Разработка модуля электронной отчетности организации позволяет сократить время, которое необходимо работнику для создания всех нужных документов. Так как вся информация находится в одном месте и легко исправляется в случае ошибки, сокращается время подписания всех документов и количество используемой бумаги.

Доступом к модулю осуществляется через браузер, а не с помощью Oracle Forms, которые требуют установки Java. Систему проще масштабировать, так как не требуется дополнительной настройки оборудования и поддержания актуальных версий программ. Процесс создания отчетов упрощается путем объединения полей, подлежащих заполнению.

Литература

1. Oracle E-Business Suite. – Режим доступа: https://ru.wikipedia.org/wiki/Oracle_E-Business_Suite.

2. Архитектура Oracle E-Business Suite R12. – Режим доступа: https://www.tadviser.ru/index.php/Продукт:Oracle_E-Business_Suite_R12

3. Oracle Application Framework (OAF). – Режим доступа: http://pressureex.narod.ru/progs/oracle_oaf/OAF_help1.html

УДК 004.93

ТЕОРЕМА О СВЁРТКЕ: СРАВНЕНИЕ ЭФФЕКТИВНОСТИ СКАНИРУЮЩЕЙ СВЁРТКИ СО СВЁРТКОЙ ЧЕРЕЗ БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ В ЗАДАЧЕ ДЕКОНВОЛЮЦИИ, С ПОМОЩЬЮ АЛГОРИТМА ЛЮСИ-РИЧАРДСОНА

И.Ю. Новоскольцев

Воронежский государственный университет

Введение

Свертка является одной из основных операций в обработке сигналов и изображений. Она позволяет комбинировать два сигнала (или изображения) вместе, чтобы получить новый сигнал, который содержит информацию о том, как каждый из исходных сигналов влияет на другой. Однако вычисление свертки прямым способом может быть очень трудоемким для больших сигналов. Здесь на помощь приходит быстрое преобразование Фурье (БПФ), которое позволяет значительно ускорить процесс вычисления свертки. В этой статье мы рассмотрим, что такое свертка через быстрое преобразование Фурье, и как их сочетание может помочь ускорить фильтрацию изображений.

1. Постановка задачи

Ставится задача деконволюции изображения f, свёрнутого фильтром g, свёрточной версией алгоритма Люси-Ричардсона, с помощью сканирующей свёртки, и свёртки с помощью быстрого преобразования Фурье.

2. Описание методов и алгоритмов

Свёртка (Convolution). В общем случае, свёртка (обозначение *) — это операция в функциональном анализе, которая при применении к двум интегрируемым относительно меры Лебега на пространстве R^n функциям f и g возвращает третью функцию, соответствующую взаимнокорреляционной функции $f(x)$ и $g(-x)$. Задача свёртки изображений заключается в применении фильтра (ядро свёртки) к изображению с целью изменения его свойств. Каждый пиксель изображения заменяется на взвешенную сумму значений пикселей изображения и соответствующих элементов фильтра, расположенных вокруг него [1, 2]. Таким образом, свёртка изображения позволяет уменьшить шум, выделить контуры, повысить резкость и др.

Формула свёртки в пространственно-временной области выглядит следующим образом:

$$(f * g) \stackrel{\text{def}}{=} \int_{R^n} f(y)g(x-y)dy = \int_{R^n} f(x-y)g(y)dy$$

Преобразование Фурье (Fourier transform, символ $\mathscr{F}$) операция, сопоставляющая одной функции вещественной переменной другую функцию вещественной переменной. Эта новая функция описывает коэффициенты («амплитуды») при разложении исходной функции на элементарные составляющие — гармонические колебания с разными частотами.

Формула дискретного преобразования Фурье следующая:

$$F(u,v) = \mathcal{F}(f(x,y)) = \int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dxdy,$$

где u и v — пространства частот.

Теорема о свертке (или Convolution theorem) в теории обработки сигналов описывает, как происходит изменение спектра сигнала после свертки двух функций. В математической формулировке теорема гласит, что свертка двух функций в пространственно-временной области эквивалентна умножению их спектров в частотной области [1, 2]. Эта теорема играет важную роль в обработке сигналов, так как позволяет эффективно применять различные фильтры и операции над сигналами, основанные на свертке.

Теорему о свёртке можно лаконично выразить следующей формулой:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}(f) \cdot \mathcal{F}(g)\},$$

где $\mathcal{F}$ — преобразование Фурье, $\mathcal{F}^{-1}$ — обратное преобразование Фурье.

Деконволюция - это процесс восстановления исходного сигнала (или изображения) из искаженной версии, которая получена путем свертки с искажающей функцией [2]. В других словах, деконволюция - это обратный процесс свертки.

Однако следует отметить, что деконволюция является вычислительно сложным процессом и может привести или потере информации при неправильном использовании. Поэтому необходимо тщательно выбирать методы деконволюции и контролировать параметры процесса восстановления для достижения наилучшего результата.

Алгоритм Люси-Ричардсона (Richardson–Lucy deconvolution) – это итерационный алгоритм восстановления изображений в задаче деконволюции, который был предложен в 1974 году Люси и в 1972 году Ричардсоном [3,4].

Суть алгоритма заключается в том, чтобы последовательно улучшать приближение к исходному изображению путем итеративной свертки текущего приближения с ядром деконволюции и вычитания разности между текущим приближением и исходным изображением из результата свертки. Таким образом, алгоритм Люси-Ричардсона позволяет оценить и восстановить исходное изображение на основе полученной деконволюции.

Алгоритм Люси-Ричардсона является одним из наиболее распространенных методов деконволюции изображений. Он может быть использован для восстановления как одномерных, так и двумерных изображений, в том числе в медицинских и научных исследованиях, где требуется точность и высокая чувствительность при обработке данных.

Формула вычисления восстановленного изображения по алгоритму Люси-Ричардсона через свёртку на шаге $t+1$ следующая:

$$\grave{f}(x,y)^{(t+1)} = \frac{f(x,y)}{\grave{f}(x,y)^{(t)} * g(-x,y)},$$

где $\grave{f}(x,y)^{(t+1)}$ — восстановленное изображение на итерации $t+1$, $f(x,y)$ — исходное изображение, $g(-x,y)$ — отражённое ядро, дробь — поэлементное деление, а начальное значение — $\grave{f}(x,y)^{(0)} = J_{m,n} \times 0.5,$ где $J_{m,n}$ — матрица единиц $m \times n$.

3. Реализация алгоритмов и вычислительный эксперимент

На рисунке 1 представлен результат применения алгоритма Люси-Ричардсона к изображению, которое было размыто Гауссовским фильтром (двухмерным Гауссианом), квадратным ядром шириной 100, дисперсией 20; после чего было восстановлено алгоритмом Люси-Ричардсона.

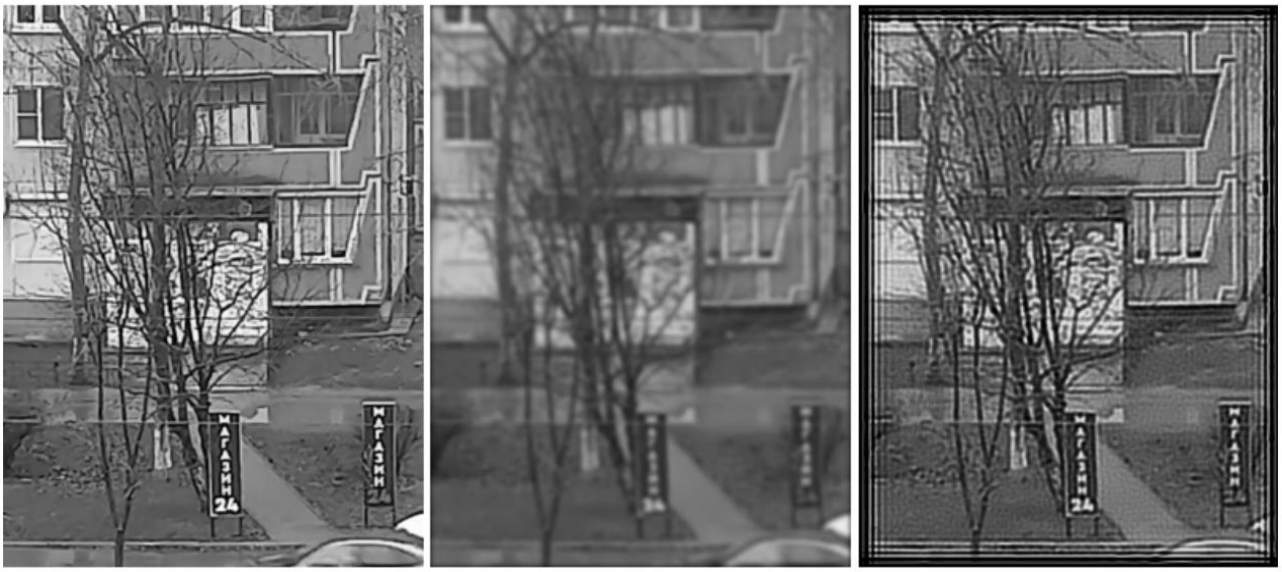


Рис. 1. Результат применения алгоритма Люси-Ричардсона: слева — исходное изображение, посередине — размытое Гауссианом и зашумлённое импульсным шумом, справа — восстановленное алгоритмом Люси-Ричардсона

Код алгоритма Люси-Ричардсона через свёртку представлен на рисунке 2, упрощённый код алгоритма представлен на рисунке 3, в этом коде отсутствует часть с центрированием изображения, это сделано для упрощения мелких деталей.

С помощью bash-сценария, зададим цикл для запуска алгоритма Люси-Ричардсона с разным количеством итераций. Результат представлен в таблице 1, способ измерения — на рисунке 4.

```
40  def lucy_richardson(image, psf, num_iter=50):
 1      deconv = np.full(image.shape, 0.5, dtype=float)
 2      psf_flipd = np.flip(psf)
 3      conv2 = fftconvolve if MANYA == False else manyaConv2  # n
 4      for _ in range(num_iter):
 5          eps = 1e-20
 6          conv = conv2(deconv, psf, mode='same') + eps
 7          relative_blur = image / conv
 8          deconv *= conv2(relative_blur, psf_flipd, mode='same')
 9      deconv[deconv > 1] = 1
10      deconv[deconv < -1] = -1
11      return deconv
```

Рис. 2. Функция алгоритма Люси-Ричардсона

```
 1
22  def manyaConv2(img, kernel, mode=None):
 1      summ = np.sum(kernel)
 2      imageFFT = rfftn(img)/summ
 3      kernelFFT = rfftn(kernel)/summ
 4      kernel = irfftn(kernelFFT).real
 5      kernelFFT = rfftn(kernel, img.shape)
 6      return (irfftn(imageFFT * kernelFFT))
 7
```

Рис. 3. Функция свёртки через быстрое преобразование Фурье

```
print(f"Output file = '{num_iter}.jpg'")
print(f'Number of iterations = {num_iter}')
start_time = time.time()
deconvolved_RL = lucy_richardson(image_noisy, psf, num_iter=num_iter)
end_time = time.time()
elapsed_time = end_time - start_time
print("Elapsed time: ", elapsed_time)
```

Рис. 2. Функция алгоритма Люси-Ричардсона

Таблица 1

Затраченное время в секундах

Количество итераций	**Свёртка сканированием**	**Свёртка через преобразование Фурье**
1	14.81	3.53
2	30.95	8.2
3	46.48	13.75
4	60.95	19.91
5	77.17	26.33
6	92.2	26.25
7	107.49	31.95
8	123.39	37.01
9	142.52	41.41
10	158.25	47.43

Результат. В результате мы получили значительное ускорение вычисления алгоритма Люси-Ричардсона через свёртку с помощью преобразования Фурье, на 10 итерациях время вычисления сократилось больше чем в 3 раза. Стоит заметить, что такое ускорение достижимо только на больших ядрах и изображениях. В случае очень маленьких ядер, свёртка сканированием может работать быстрее.

К сожалению, автор этой работы не смог вывести точную формулу для вычисления вычислительной сложности двух алгоритмов свёртки, но ориентировочно можно оценить через нотацию большого О как: преобразование Фурье имеет сложность $O(N\log N)$, операция поэлементного умножения — $O(N)$, обратного преобразования Фурье — тоже $O(N\log N)$, следовательно, вычислительную сложность свёртки через БПФ можно оценить как $O(MN\log MN)$. Вычислительную сложность сканирующей свёртки можно оценить как $O(MNkk)$, где $M \times N$ — разрешение изображения, а $k \times k$ — разрешение ядра свёртки.

Заключение

В этой работе был опробован метод ускорения деконволюции изображений алгоритмом Люси-Ричардсона с помощью быстрого преобразования Фурье. Было оценено время выполнения алгоритма через быстрое преобразование Фурье и сканирующую свёртку. Был получен ожидаемый результат, алгоритм был значительно ускорен.

Работа выполнена под руководством канд. физ.-мат. наук, доцента кафедры вычислительной математики и прикладных информационных технологий Воронежского государственного университета Королькова Олега Геннадьевича. E-mail: korolkov@amm.vsu.ru.

Литература

1. О'Нейл Э. Введение в статистическую оптику / Э. О'Нейл – Москва : Мир, 1966. – 255 с.

2. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс – 3-е изд. – Москва : Изд-во Техносфера, 2018. – 1104 с.

3. Richardson W. H. Bayesian-Based Iterative Method of Image Restoration / W. H. Richardson // Journal of the Optical Society of America. – 1972. – Т.62. – С. 55–59.

4. Новейшие методы обработки изображений / А.А. Потапов [и др.]; под ред. А.А. Потапов – Москва : Физматлит, 2016. – 498 с.

УДК 519.6

ИССЛЕДОВАНИЕ КРИТЕРИЕВ И ПРАВИЛ ВЫБОРА ОБРАЗОВАТЕЛЬНЫХ НАПРАВЛЕНИЙ МЕТОДАМИ СТАТИСТИЧЕСКОГО АНАЛИЗА ДАННЫХ И МАШИННОГО ОБУЧЕНИЯ

В. Р. Осипова

Воронежский государственный университет

Введение

В условиях цифровой экономики очень сильно возрастает роль образования, которое должно готовить специалистов, имеющих необходимые ключевые и профессиональные компетенции. Федеральные стандарты последнего поколения тесно связаны с профессиональными стандартами, отражающими обобщенные компетентностные модели требуемых в различных сферах специалистов, формулируют цели, задачи и компетентностное содержание для каждого образовательного направления. Федеральные стандарты находятся в открытом доступе, и каждый абитуриент, осуществляющий свой выбор на рынке образовательных услуг, может ознакомиться с ними. Абитуриенты выбирают для себя направление подготовки исходя из своих предпочтений, степени информированности в предметной области и оценки доступности анализируемых вариантов выбора. В условиях цифровой среды ВУЗы предоставляют открытую, достаточно полную информацию о направлениях подготовки. Ведется серьезная профориентационная работа. Кроме того, аналитические агентства ежегодно предоставляют результаты исследований по востребованности на рынке труда различных специалистов, и о новых компетенциях, которые необходимы для успешного трудоустройства.

В рамках данной статьи ставится задача предложить структуру анализа, который базировался бы на применение методов статистического анализа данных и машинного обучения и который на основе данных анкетирования студентов различных образовательных направлений позволил бы выявить, какие критерии при современном уровне информированности используют респонденты при выборе вузов и образовательных направлений магистратуры.

Анализ критериев и правил выбора позволит: для каждого вуза построить радарную диаграмму критериев выбора и на их основе провести сравнительный анализ, выявить наиболее типичные траектории продолжения образования в магистратуре, выявить новые нетрадиционные критерии и правила выбора, прогнозировать возможный дисбаланс на рынке труда, оценить какие механизмы профориентационной работы могут поменять предпочтения в выборе.

1. Материалы

В основе проводимого исследования лежат результаты анкетирования магистров, обучающихся в различных вузах, на различных группах специальностей и на различных магистерских программах. Всего данный опрос проходили 1100 студентов. Он состоит из различного рода вопросов, касающихся непосредственного обучения студентов. В том числе анкета включает в себя следующие основные пункты: пол, возраст, материальное положение семьи, условия обучения (бесплатно / платно), образование родителей, регион проживания, цель получения образования, полнота информированности о современных тенденциях на

рынке труда, образование на момент поступления в магистратуру, субъективные причины выбора ВУЗа, в котором респондент обучается в данный момент времени, субъективные причины решения о поступлении в магистратуру, субъективные причины выбора направления, на котором респондент обучается в данный момент времени, приоритеты в будущей работе.

2. Статистический анализ результатов анкетирования для выявления причин выбора направлений обучения в магистратуре

Анализ данных анкетирования проводился с помощью библиотек языка программирования Python.

2.1. Выявление основных причин, влияющих на решение поступать в магистратуру

Для выявления основных причин, влияющих на решение респондентов поступать в магистратуру, была построена гистограмма, представленная на рисунке 1.

Рис. 1. Гистограмма причин, влияющих на решение поступать в магистратуру

Наиболее частым ответом на вопрос «По каким причинам вы решили поступить в магистратуру?” является «Чтобы иметь лучшие возможности для карьерного роста». Вузы, предоставляющие образовательные услуги, должны учитывать важность этого критерия, стремиться к достижению целей абитуриентов по данному критерию, формированию потенциала карьерного роста, тщательно планируя связи с работодателями и организацию мероприятий по формированию будущей карьеры обучающихся.

Большое число респондентов также отмечает в качестве причин, повлиявших на их решение поступать в магистратуру, возможность получения образования по другому направлению подготовки, самореализацию и саморазвитие, а также возможность получать более высокую заработную плату. При планировании учебного плана магистратуры, должна учитываться возможность обучения абитуриентов, имеющих подготовку по близким направлениям бакалавриата, так чтобы обучение действительно являлось развитием и обеспечивало синергетический эффект с имеющимися знаниями, умениями и навыками.

2.2. *Выявление основных причин выбора направлений обучения в магистратуре*

Для выявления основных причин выбора направлений обучения в магистратуре были рассмотрены следующие укрупненные группы направлений — гуманитарные науки;

образование и педагогические науки; науки об обществе; инженерное дело, технологии и технические науки; математические и естественные науки.

Респондентам предлагалось ответить на вопрос «По каким причинам вы выбрали направление подготовки, по которому вы обучаетесь?» со следующими вариантами ответов:

1. Выбрал(а) по совету родителей / друзей / знакомых;
2. Дает возможность карьерного роста;
3. Дает возможность легко найти работу в нашей стране;
4. Дает возможность легко найти работу за рубежом;
5. Дает возможность хорошо зарабатывать;
6. Дает социально значимую профессию, которая приносит пользу людям, государству;
7. За компанию с друзьями;
8. На это направление подготовки было легче поступить;
9. По этому направлению подготовки легко учиться;
10. По этому направлению подготовки невысокая оплата за обучение / бесплатное обучение;
11. По этому направлению подготовки учился(лась) на бакалавриате;
12. По этому направлению работают мои родственники / знакомые;
13. Подавал(а) еще на другие направления подготовки, но удалось поступить только на это;
14. Позволит иметь удобный график работы;
15. Позволит иметь хорошее социальное обеспечение на работе;
16. Позволит иметь хорошие условия труда;
17. Позволит получить интересную и разнообразную работу;
18. Престижное направление подготовки;
19. Соответствует моим способностям и интересам;
20. Это был случайный выбор;
21. Другое.

Для визуализации причин выбора направления обучения в магистратуре и сравнения показателей по укрупненным группам направлений построим радарные диаграммы (рис. 2 – рис. 6).

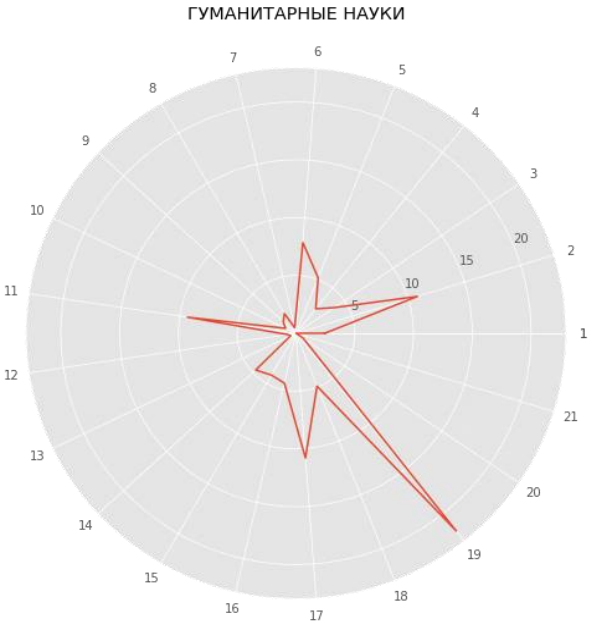


Рис. 2. Радарная диаграмма критериев выбора с процентными показателями для направления «Гуманитарные науки»

Основными причинами выбора гуманитарного направления являются соответствие способностям и интересам респондента, возможности карьерного роста и получения интересной и разнообразной работы.

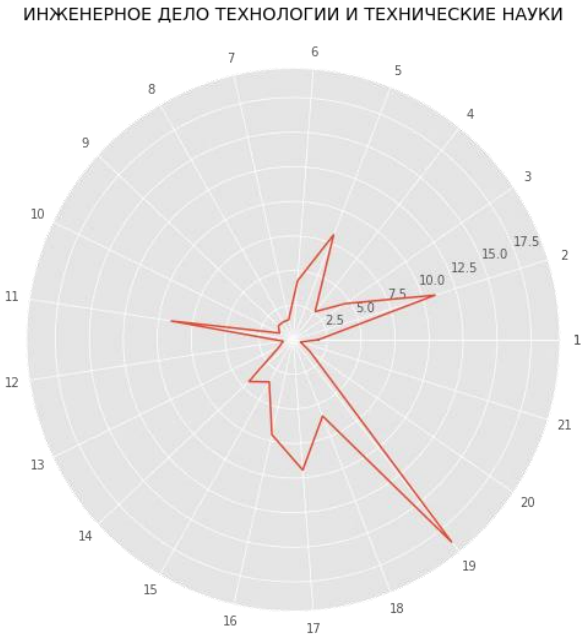


Рис. 3. Радарная диаграмма критериев выбора с процентными показателями для направления «Инженерное дело, технологии и технические науки»

Основными причинами выбора направления «Инженерное дело, технологии и технические науки» являются соответствие способностям и интересам респондента, возможности карьерного роста и получения интересной и разнообразной работы.

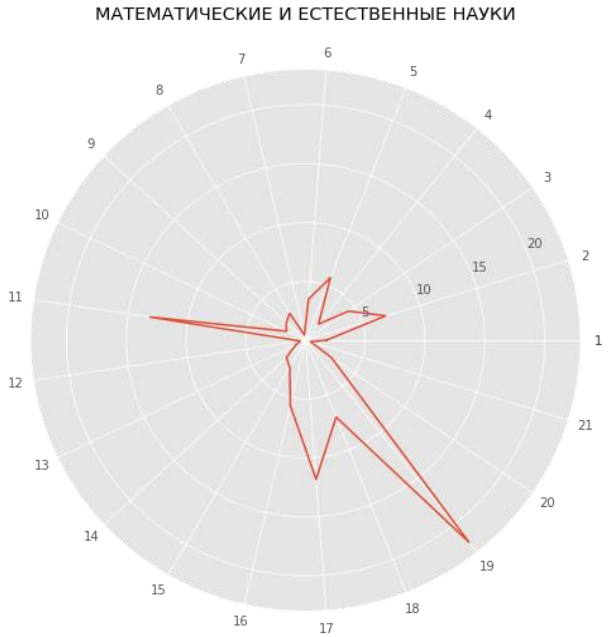


Рис. 4. Радарная диаграмма критериев выбора с процентными показателями для направления «Математические и естественные науки»

Основными причинами выбора направления «Математические и естественные науки» являются соответствие способностям и интересам респондента и возможность получения интересной и разнообразной работы. Кроме того, на данном направлении обучения респонденты чаще, чем на других, указывают в качестве причины возможность продолжения обучения по тому же направлению, что и в бакалавриате.

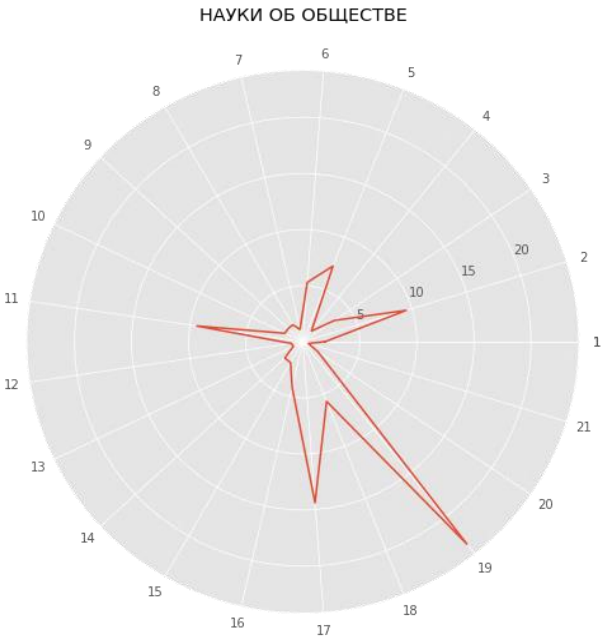


Рис. 5. Радарная диаграмма критериев выбора с процентными показателями для направления «Науки об обществе»

Основными причинами выбора направления «Науки об обществе» являются соответствие способностям и интересам респондента и возможность получения интересной и разнообразной работы.

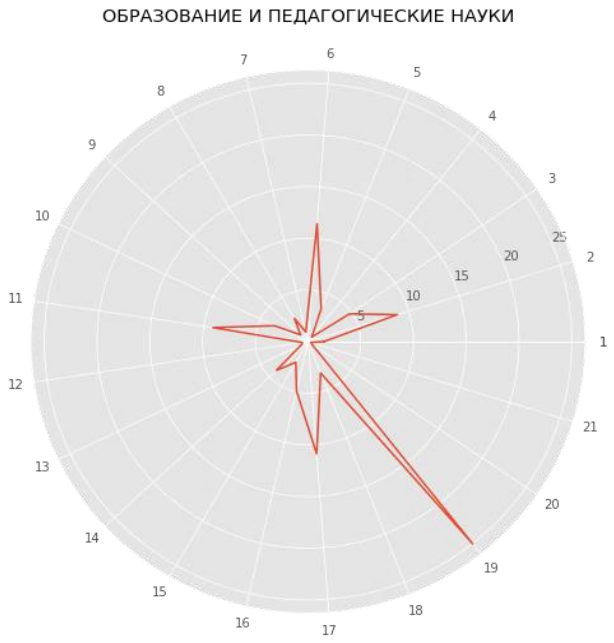


Рис. 6. Радарная диаграмма критериев выбора с процентными показателями для направления «Образование и педагогические науки»

Основными причинами выбора направления «Образование и педагогические науки» являются соответствие способностям и интересам респондента и возможность получения интересной и разнообразной работы. Также на данном направлении респонденты выделяют возможность получения социально значимой профессии, которая приносит пользу людям и государству.

Таким образом, самыми важными критериями при выборе любого направления магистратуры являются соответствие способностям и интересам и возможность получения интересной и разнообразной работы, что может объяснятся более осознанным подходом к выбору будущей профессии, чем в бакалавриате.

3. Применение методов машинного обучения для предсказания укрупненных групп специальностей

В качестве метода прогнозирования был выбран наиболее эффективный метод — градиентный бустинг на базе деревьев классификации.

Градиентный бустинг позволяет делать прогнозы на основе ансамбля слабых обучающих алгоритмов. Он представляет собой процедуру последовательного построения композиции алгоритмов машинного обучения, при котором каждый последующий алгоритм пытается устранить недостатки композиции предыдущих алгоритмов.

Для данного метода отмечается почти неограниченное уменьшение частоты ошибок на независимых тестовых данных по мере построения композиции. Вместе с тем, качество на тестовых данных зачастую продолжает расти даже после получения для всей обучающей выборки безошибочного распознавания.

На рисунке 7 представлены признаки, которые оказывают на построенную модель предсказания укрупненной группы специальностей наибольшее влияние.

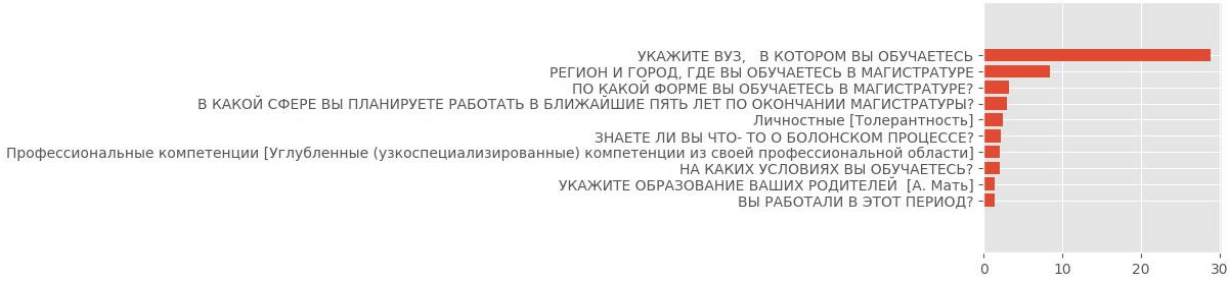


Рис. 7. Гистограмма важности признаков модели градиентного бустинга

Наибольшее значение имеет непосредственно направленность самого ВУЗа, где респондент обучается, так как большинство ВУЗов в России чаще являются узконаправленными, чем готовящими специалистов различных направлений. Также важными признаками являются регион и город, где происходит обучение; форма обучения — так как на некоторых направлениях возможно только очное обучение; планируемая сфера работы в ближайшие 5 лет после окончания магистратуры; сформированность некоторых знаний, умений и навыков, характерных для определенных направлений.

Заключение

В данном исследовании был проведен анализ, позволяющий на основе данных анкетирования студентов различных образовательных направлений выявить основные критерии, которые при современном уровне информированности используют респонденты при выборе вузов и образовательных направлений магистратуры. Была построена модель, предсказывающая по данным анкетирования укрупненную группу специальностей, по которой осуществляется обучение. Были выделены наиболее важные критерии, влияющие при выборе направления магистратуры — соответствие способностям и интересам, возможность получения интересной и разнообразной работы, и основные признаки, указывающие на само направление обучения в магистратуре — направленность ВУЗа, регион и город места обучения, форма обучения, планируемая сфера работы в ближайшие 5 лет после окончания магистратуры, сформированность определенных знаний, умений и навыков.

Литература

1. Разработка методики анализа эффективности магистерских программ: коллективная монография / под ред. Е. А. Сухановой; Нац. исслед. Томский гос. ун-т. – Томск: 2019. – 109 с.

2. Дудин М.Н., Шкодинский С.В., Вашаломидзе Е.В. Цифровая экономика: новые вызовы и возможности для рынка труда и высшего образования в России // Экономика труда. – 2021. – Том 8. – № 10. – С. 1089-1104.

3. Чубукова, И. А. Data Mining : учебное пособие / И. А. Чубукова. – 2-е изд. – Москва : ИНТУИТ, 2016. – 470 с. – ISBN 978-5-94774-819-2. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/100582 – (Дата обращения: 05.04.2023)

УДК 519.6; 658.5

ОДИН ИЗ ПОДХОДОВ ФОРМИРОВАНИЯ ЭФФЕКТИВНЫХ КОМАНД СТАРТАП-ПРОЕКТОВ

С.А. Палкина, А. Э. Потемкина

Воронежский государственный университет

Введение

В условиях большого объема предложений кандидатов на вакансии в ИТ проектах тема подбора специалистов становится особенно актуальной. С одной стороны, руководство стремиться отбирать специалистов лучших в профессиональном смысле, с другой стороны, успешность стартап-проектов определяется эффективностью работы группы сотрудников в целом, а не отдельными сотрудниками, и отдается предпочтение личностным характеристикам. Однако, ни в одном из этих вариантов не учитывается мотивация сотрудника и уровень работоспособности.

В работе поставлена цель: разработать модель оценки кандидатов для отбора в команды стартап-проектов в сфере ИТ, которая позволит учитывать профессиональные, личностные характеристики, потенциал работы. Это позволит формировать команды из специалистов, которые могут решать задачи и профессионально, и коллективно, и в «жестких» условиях старап-проектов сферы ИТ.

1. Особенности формирования команд для стартап-проектов

1.1. Постановка задачи

С позиции руководителя проекта, команда рассматривается как коллектив, который обладает необходимыми для проекта профессиональными компетенциями и способен в кратчайший срок обеспечить реализацию проекта. В работах Р.М. Белбина, В.В. Богданова и М. Разу роль команды в обеспечении успешности проекта является определяющей. Соответственно, помимо профессиональных качеств, участники должны обладать рядом личностных качеств, позволяющих работать в коллективе и обеспечивать высокую работоспособность. Сделаем еще одно уточнение, главной характеристикой стартап-проекта является наличие потенциала для стремительного роста. Соответственно, команда для создания успешного стартап-проекта должна быть эффективной. Под эффективностью понимается достижение высоких результатов с наименьшими затратами, в том числе и по времени. В связи с этим, при подборе членов команды следует учитывать и потенциал кандидата в столь специфичной работе. Ставится задача: сформировать модель оценки кандидатов в проект, которая позволит учесть личностные, профессиональные характеристики кандидата, а также потенциал для эффективной работы.

1.2. Процесс формирования команды в стартап-проект

Процесс формирования команды стартап-проекта может быть представлен в виде следующих семи шагов.

1. Руководство компании собирает необходимую информацию о стартап-проекте и формирует портрет идеального руководителя.

2. Формируется экспертная комиссия для определения руководителя проекта. Экспертной комиссией формируется перечень необходимых характеристик портрета идеального руководителя и диапазон допустимых значений по ним.

3. С учетом требований/допустимых значений характеристик формируется база кандидатов на основе внутреннего банка сотрудников и предложений, поступивших из внешних источников по подбору персонала. При необходимости банк кандидатов расширяется.

4. Экспертная комиссия проводит оценку кандидатов и утверждает руководителя проекта.

5. Руководителем определятся процедура набора команды, формируется видение рабочей группы и отдельных ее участников, устанавливаются критерии для отбора членов команды.

6. По данным критериям формируется база кандидатов. При отсутствии необходимого(ых) кандидата(ов) база расширяется.

7. Руководитель проводит оценку кандидатов, согласовывает их участие и формирует команду проекта.

Для оценки кандидатов в проект предлагается модель оценки кандидатов.

2. Модель оценки кандидатов

Модель оценки позволяет сформировать видение идеального кандидата в проект и упростить процесс оценки потенциального кандидата. Модель сформирована согласно методу многокритериальной оптимизации – методу анализа иерархий (МАИ), соответственно, позволяет одновременно учитывать множество критериев. Метод МАИ основан на декомпозиции проблемы (цели) на составляющие, упрощающие выбор. В данном случае целью является идеальный кандидат. Составляющими декомпозиции являются критерии оценки кандидата. Последний уровень модели представляет собой ответы на вопрос «Что нужна знать о кандидате, чтобы выбрать его?» Это информация о кандидате, которая является характеристиками кандидата. Согласно МАИ, на базе модели строится видение кандидата, которое заключается в оценке руководителем проекта (лицом принимающем решение) критериев отбора кандидата. Оценки критериев позволяют рассчитать веса характеристик кандидата и по ним ранжировать претендентов. Модель для формирования портрета идеального кандидата и оценки соискателей представлена на рис. 1.

Согласно модели, отбор и оценка каждого потенциального кандидата производится по таким показателям как профессиональные и личностные характеристики, статусу, условиям работы и ожиданиям. В профессиональных характеристиках учитываются образование, опыт работы, знания, умения и навыки (ЗУН). Предполагается, что ЗУН оцениваются отдельно. При оценке кандидатов используется итоговый балл, полученный, например, по итогу тестирования. Относительно личностных характеристик принимаются во внимание возраст, черты характера, поведенческая модель, возможная командная роль. Личностные характеристики могут быть получены как результат самооценки кандидата, так и по результатам отдельного тестирования.

Компонент «Статусы, ожидания, условия работы» позволяет сделать выбор между кандидатом более или менее загруженным в других проектах, готовым к работе в офисе или согласным только на частичную занятость и дома, потенциально загруженным в периоды сессий или свободным, но ожидающим большей заработной платы. Это связано с тем, что для хорошей работоспособности важны и удаленность работы от дома, в случае офисной занятости, и небольшая нагрузка в других проектах, и образ жизни, который зависит от того, соискатель студент или молодой несемейный человек, и даже от ожиданий по заработной плате.

Рис.1. Модель оценки соответствия кандидата вакансии

Характеристики нижнего уровня модели считаются заданными на момент оценки кандидатов.

Заключение

Модель является обобщенной и ориентирована на отбор специалистов стартап-проектов в ИТ сфере. Профессиональные знания, умения и навыки заложены отдельным элементом и, в зависимости функциональных обязанностей по проекту, могут быть любыми. То же относительно личностных характеристик. Универсальность модели заключается в том, что при формировании команды одного проекта, руководителю не обязательно формировать видение для каждого должности. Достаточно задать оценку только по уровню элементов, которые разняться, например, по «знания, умения и навыки» или «Личностные характеристики».

Литература

1. Азарнова Т.В. Модели и методы принятия решений / Составители: Т.В. Азарнова, Ю.В. Бондаренко, Н.Б. Баева, Е.С. Дашкова, В.В. Ухлова; Воронежский государственный университет. – Воронеж: Издательский дом ВГУ, 2021. – 310 с.

2. Бондаренко Ю.В., Горошко И.В., Васильчикова Е.В. Экспертно-тестовый механизм комплексной оценки кандидатов при подборе персонала // Вестник Южно-Уральского государственного университета. Серия: Компьютерные технологии, управление, радиоэлектроника. 2020. Т. 20. № 1. – С. 100-110.

3. Джабраилова, З.Г. Моделирование процесса выбора кандидатов на вакантные должности с применением нечеткой логики / З.Г. Джабраилова, С.Р. Нобари // Искусственный интеллект: сб. статей. – Баку, 2009. – С. 254–259.

4. Жуков, Ю. М., Журавлёв А. В., Павлова Е. Н. Технологии командообразования: учеб. пособие для студ. вузов Москва : Аспект Пресс, 2008. – 320 с.

5. Йеттер, В. Эффективный отбор персонала. Метод структурированного интервью / В. Йеттер. – Харьков : Изд-во «Гуманитарный центр», 2011. – 360 с.

6. Кибанов, А.Я. Управление персоналом организации: стратегия, маркетинг, интернационализация: учеб. пособие / А.Я. Кибанов, И.Б. Дуракова. – Москва : Инфра-М, 2009. – 301 с.

7. Магура, М.И. Поиск и отбор персонала. – Москва, 2003. – 312 с.

8. Ногин, В.Д. Принятие решений при многих критериях / В.Д. Ногин. – Санкт-Петербург : ЮТАС, 2007. – 104 с.

9. Саати, Т. Принятие решений. Метод анализа иерархий / Т. Саати., – пер. с англ. – Москва : Радио и связь, 1993. – 278 с.

10. Ухлова, В. В. Применение системного подхода в процедурах формирования команд стартапов ИТ-проектов / Актуальные проблемы прикладной математики, информатики и механики / В. В. Ухлова, С. А. Палкина // Сборник трудов Международной научной конференции, Воронеж, 13-15 декабря 2021 г. — Воронеж, 2022. — С. 791-796.

11. Тележенкова, М. Д. Оценка факторов эффективности стартапов / М.Д. Тележенкова, Е. О. Забкова // Экономика и бизнес: теория и практика. 2018. №12-2. URL: https://cyberleninka.ru/article/n/otsenka-faktorov-effektivnosti-startapov.

УДК 537.9

МОДЕЛИРОВАНИЕ ТЕРМОДИНАМИЧЕСКИХ ХАРАКТЕРИСТИК АНСАМБЛЯ СЕГНЕТОЭЛЕКТРИЧЕСКИХ НАНОЧАСТИЦ В ОКРЕСТНОСТИ ТОЧКИ ФАЗОВОГО ПЕРЕХОДА ВТОРОГО РОДА

И. А. Панков, Д. Р. Юнда, А. В. Шуба

ВУНЦ ВВС «ВВА им. проф. Н.Е. Жуковского и Ю.А. Гагарина» (г. Воронеж)

Введение

Как известно, свойства наноразмерных объектов существенно отличаются от свойств объемных материалов [1, 2] в связи с существенной зависимостью распределения параметра порядка от координат в ограниченных образцах. Определить ее даже в простом случае пренебрежения собственными полями рассеяния аналитически достаточно сложно. Использование же различных численных методов решения (метод конечных разностей [3], метод конечных элементов [4]) в случае большого количества нанообъектов требуют серьезных вычислительных ресурсов и сопряжено с погрешностями вычислений. Применение вариационного метода [5] позволяет выбрать пробную функцию по наименьшему значению свободной энергии и продвинуться аналитически в решении подобного класса задач, однако, найденное таким способом распределение не может гарантированно являться истинным распределением, а перебор всех возможных профилей параметра порядка для установления реального его значения не представляется возможным. Поэтому для аналитического исследования подобных объектов прибегают к различного рода упрощениям, одним из которых является предложенная в данной работе кусочно-параболическая аппроксимация свободной энергии наночастицы с сохранением ее нелинейности.

1. Постановка задачи

Целью настоящей работы является определение температурных зависимостей теплоемкости и диэлектрической проницаемости ансамбля сегнетоэлектрических наночастиц с учетом неоднородного распределения поляризации внутри них.

Для определения координатной функции поляризации сегнетоэлектрической наночастицы использовался численно-аналитический подход на основе кусочно-параболической аппроксимации ее свободной энергии, подробно изложенный в [6]. Для отдельной частицы, предположительно имеющей сферическую симметрию радиусом R и претерпевающей фазовый переход второго рода, свободная энергия имеет вид [7]

$$F = \int_V \left\{ F_\infty + \frac{\kappa}{2}\left(\nabla P\right)^2 - \mathbf{E}_{ext}\mathbf{P} \right\} dV + \int_S \frac{\alpha_s}{2}\mathbf{P}^2 dS, \qquad (1)$$

где $\mathbf{P} = \left\{0, 0, P\left(\rho\right)\right\}$ – вектор поляризации, играющий роль параметра порядка; $\mathbf{E}_{ext}$ – вектор напряженности внешнего электрического поля; $\kappa \sim a^2$ корреляционная постоянная, где a – межатомное расстояние; α_s – коэффициент при квадратичном слагаемом в поверхностной энергии; F_∞ – однородная часть свободной энергии, равная

$$F_\infty = -\frac{\alpha}{2}\mathbf{P}^2 + \frac{\beta}{4}\mathbf{P}^4, \qquad (2)$$

где $\alpha = \alpha_0 \left(T_C - T \right)$, β – коэффициенты в разложении свободной энергии вблизи температуры Кюри T_C объёмного образца. Минимизация функционала (1) дает уравнение равновесия

$$\kappa \Delta P + \alpha P - \beta P^3 + E_{ext} = 0 \tag{3}$$

с граничным условием общего типа

$$\kappa \nabla P + \alpha_s P \big|_{\rho = R} = 0. \tag{4}$$

Для упрощения краевой задачи (3), (4) с сохранением ее нелинейности аппроксимируем однородную часть свободной энергии (2) системой парабол с вершинами $\left(0,0\right)$ и $\left(\pm P_\infty, -\alpha P_\infty^2 / 3 \right)$, гладко «сшиваемых» в точках $P^* = \pm 2P_\infty / 3$ (равенство функций и их производных), где $P_\infty = \sqrt{\alpha / \beta}$ – поляризация объемного кристалла. Такая аппроксимация функционала (1) позволяет заменить нелинейное уравнение (3) двумя линейными уравнениями с решениями в безразмерных переменных

$$\tilde{P}^{\mathrm{I}} \left(\tilde{\rho} \right) = C_1 \frac{\mathrm{I}_{1/2} \left(\sqrt{2\alpha} \tilde{\rho} \right)}{\sqrt{\tilde{\rho}}} + 1 + \frac{\tilde{E}_{ext}}{2\alpha}, \quad \tilde{\rho} \le \tilde{\rho}^*; \tag{5}$$

$$\tilde{P}^{\mathrm{II}} \left(\tilde{\rho} \right) = C_2 \frac{\mathrm{J}_{1/2} \left(\sqrt{\alpha} \tilde{\rho} \right)}{\sqrt{\tilde{\rho}}} + C_3 \frac{\mathrm{N}_{1/2} \left(\sqrt{\alpha} \tilde{\rho} \right)}{\sqrt{\tilde{\rho}}} - \frac{\tilde{E}_{ext}}{\alpha}, \quad \tilde{\rho}^* < \tilde{\rho} \le \tilde{R} \tag{6}$$

для внутренней и внешней областей наночастицы, гладко «сшиваемые» в точке ρ^*. Здесь и далее величины с размерностью поляризации измеряются в единицах P_∞, с размерностью длины – в межатомных расстояниях *a*. В решениях (5), (6) обозначения $\mathrm{I}_{1/2}\left(x\right)$, $\mathrm{J}_{1/2}\left(x\right)$ и $\mathrm{N}_{1/2}\left(x\right)$ соответствуют функциям Инфельда, Бесселя и Неймана порядка ½ соответственно [8]. Константы интегрирования C_1, C_2, C_3 и точку $\tilde{\rho}^*$ определяем численно из условий гладкой «сшивки» функций (5), (6) с учетом граничного условия (4) для функции (6). Для расчетов используем параметры триглицинсульфата (ТГС): T_C=322 К, α_0=3.92·10^{-3} К$^{-1}$, β=8·10^{-10} ед. СГСЭ [9].

2. Решение задачи и анализ результатов

2.1. Ионный вклад в диэлектрическую проницаемость

Зависимость поляризации от внешнего электрического поля E_{ext} , присутствующая в формулах (5), (6), позволяет найти вклад отдельных гранул в диэлектрическую проницаемость ε_s их ансамбля

$$\varepsilon_s \left(T \right) = \int_0^{\tilde{R}_s + 3\sigma} \varepsilon \left(\tilde{R}, T \right) \cdot f \left(\tilde{R} \right) d\tilde{R}, \tag{7}$$

с учетом распределения $f\left(\tilde{R}\right)$ гранул по размерам, где $\varepsilon\left(\tilde{R}, T\right)$ – диэлектрическая проницаемость частицы с радиусом $\tilde{R} = \tilde{R}_i$ при температуре *T*, определяемая при наложении малого внешнего поля E_{ext} как

$$\varepsilon\left(\tilde{R}, T\right) \approx \frac{E_{ext}}{P\left(E_{ext}\right) - P\left(0\right)}. \tag{8}$$

При отсутствии информации о виде распределения $f\left(\tilde{R}\right)$, как правило [10, 11], ограничиваются нормальным законом:

$$f\left(\tilde{R}\right)=\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(\tilde{R}-R_s)^2}{2\sigma^2}\right) \tag{9}$$

с характерным (средним) радиусом частиц R_s и среднеквадратическим отклонением σ. На рис. 1 представлены температурные зависимости вклада в диэлектрическую проницаемость ε_s ансамбля сегнетоэлектрических частиц ТГС, рассчитанные численно по формуле (7) с учетом выражений (8), (9).

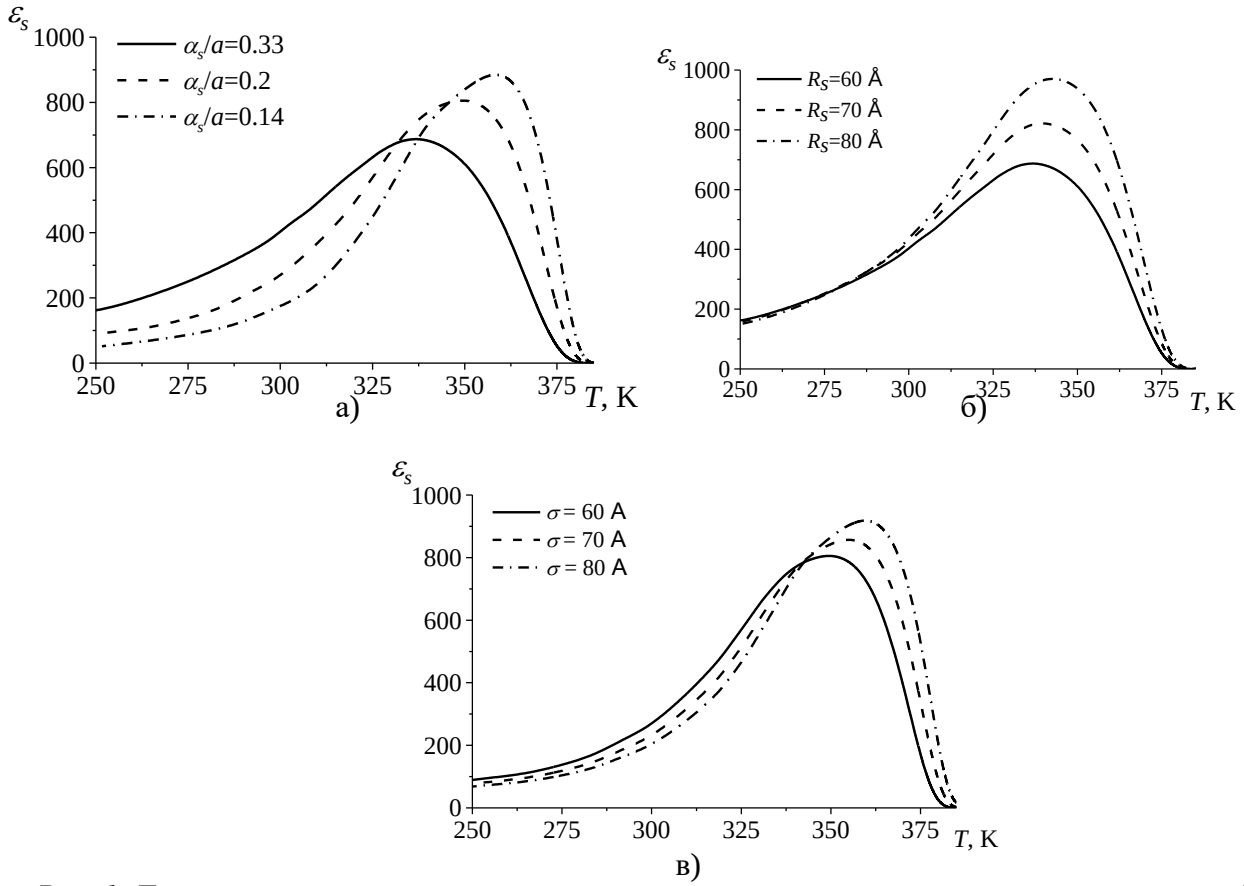


Рис. 1. Температурные зависимости вклада в диэлектрическую проницаемость ε_s ансамбля частиц: а) R_s=60 Å, σ =60 Å; б) α_s=0.33a, σ =60 Å; в) R_s=60 Å, α_s=0.2a

Аналогичным образом можно рассчитать вклад в теплоемкость $c_p\left(T\right)$ ансамбля сегнетоэлектрических частиц (рис. 2), определяемый теплоемкостью каждой из них в виде

$$c_p\left(T,\tilde{R}\right)=-T\frac{\partial^2 F}{\partial T^2}, \tag{10}$$

зависящей от значения F свободной энергии (1).

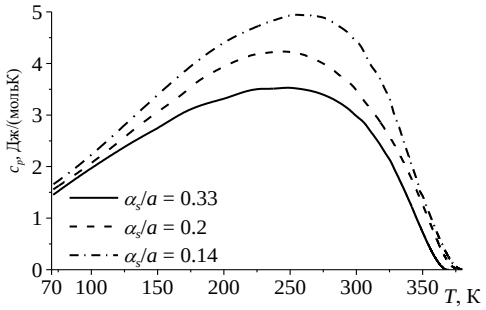


Рис. 2. Температурные зависимости вклада в теплоёмкость c_p ансамбля частиц с характерным радиусом R_s=60 Å и среднеквадратическим отклонением σ =60 Å

Существенная зависимость локальной температуры фазового перехода от размера и степени «закрепления» поляризации на границе наночастиц приводит к различному положению максимумов T_{max} диэлектрической проницаемости ε_s и теплоёмкости c_p на температурной шкале (рис. 1, рис. 2). Уменьшение степени «закрепления» α_s поляризации на границе частиц увеличивает их поляризацию, а, следовательно, и диэлектрическую проницаемость (рис. 1а). Поляризация и диэлектрическая проницаемость частиц $\varepsilon\left(\tilde{R},T\right)$ также растут с увеличением их размера, поэтому с ростом характерного радиуса R_s частиц диэлектрический отклик $\varepsilon_s\left(T\right)$ ансамбля увеличивается (рис. 1б). Максимум диэлектрической проницаемости T_{max} смещается в сторону более высоких температур, т.к. с увеличением размера частиц необходимо меньшее переохлаждение ниже T_C для появления в них полярного состояния. Увеличение среднеквадратического отклонения σ повышает долю крупных частиц в ансамбле, что усиливает диэлектрический отклик и смещает T_{max} вверх по температурной шкале (рис. 1в).

2.2. Суперпараэлектрический вклад в диэлектрическую проницаемость

Помимо вклада в диэлектрическую проницаемость, связанного с деформацией частиц в поле, существует вклад, предсказываемый классической теорией Ланжевена, обобщённой на случай частиц конечных размеров – суперпараэлектричество [12, 13]. Часть частиц, у которых свободная энергия сравнима с тепловой энергией k_BT, при наложении поля E_{ext} могут переключаться, т.е. переориентироваться по направлению поля как единое целое.

Если принять, что выгодные (по полю) и невыгодные (против поля) ориентации частиц подчиняются статистике Больцмана, последние могут давать вклад в диэлектрическую проницаемость [14]:

$$\varepsilon_T\left(T\right)=\sum_i\frac{2\pi\mathcal{P}_i^2N_i}{3k_BT}, \tag{11}$$

где N_i – объемная концентрация таких частиц, существенно зависящая от их размера и температуры. Здесь важно учитывать, что в частицах сверхмалого размера поляризация может возникнуть только в случае отрицательного значения свободной энергии (1), когда объемная составляющая F_∞ больше по модулю суммы неоднородной и поверхностной составляющих (рис. 3).

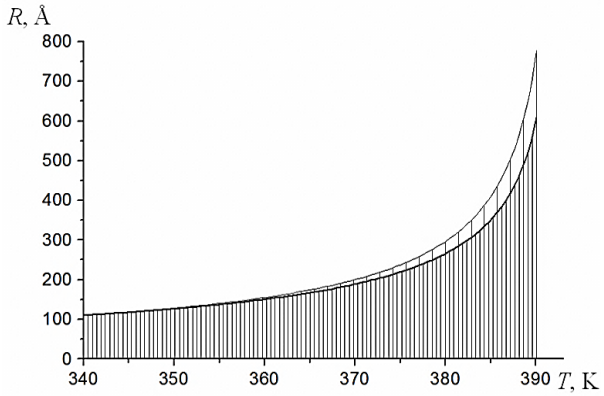


Рис. 3. Температурные зависимости размеров частиц, способных переключаться как единое целое (крупная штриховка) и размеров частиц, ниже которых сегнетоэлектрическая фаза отсутствует (мелкая штриховка) при $\alpha_s = 0.33a$

Подставляя выражение (11) в формулу (7) с учетом нормального закона распределения (9), получаем температурные зависимости вклада в диэлектрическую проницаемость от частиц, способных переключаться как единое целое (рис. 4).

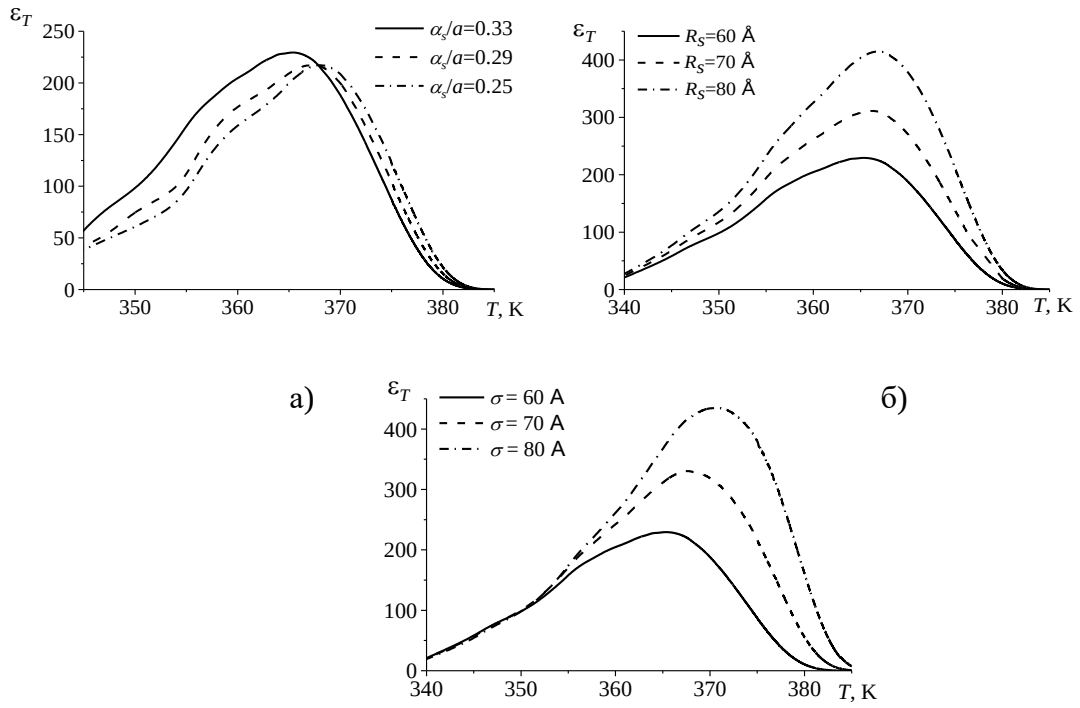


Рис. 4. Температурные зависимости вклада в диэлектрическую проницаемость ε_T:
а) R_s=60 Å, σ =60 Å; б) $\alpha_s = 0.33a$, σ =60 Å; в) $\alpha_s = 0.33a$, R_s=60 Å

Из рис. 4 видно, что, не смотря на сравнительно малую долю частиц, меняющих направление поляризации на противоположное (рис. 3), их вклад в диэлектрическую проницаемость сравним с ионным вкладом, найденным в п. 2.1.

Заключение

На основе теории Ландау – Гинзбурга с применением кусочно-параболической аппроксимации свободной энергии проведено моделирование термодинамических

характеристик ансамбля сегнетоэлектрических наночастиц из ТГС. Определены теплоемкость, ионный и суперпараэлектрический вклады в диэлектрическую проницаемость ансамбля. Показано, что широкий интервал температур фазового перехода наноразмерных частиц может служить одной из основных причин размытия термодинамических свойств по температуре в неоднородных материалах типа гранулированных нанокомпозитов сегнетоэлектрик-диэлектрик или сегнеторелаксоров. Величина и положение температурных максимумов физических свойств таких материалов в существенной мере зависят от распределения частиц по размерам: рост характерного радиуса R_s повышает отклик материала на внешнее электрическое поле. Величина суперпараэлектрического диэлектрического отклика сравнима с ионным вкладом в диэлектрическую проницаемость от достаточно крупных частиц.

Литература

1. Физика сегнетоэлектриков: современный взгляд / под ред. К. М. Рабе, Ч. Г. Ана, Ж.-М. Трискона. – Москва: Лаборатория знаний, 2020. – 443 с.
2. Нечаев, В. Н. Размерные эффекты в фазовых переходах и физических свойствах ферроиков: монография / В. Н. Нечаев, А. В. Шуба. – Москва: ИНФРА-М, 2023. – 384 с.
3. Cui, L. Study on the dynamic critical behavior of a ferroelectric heterostructure / L. Cui, C. Chen, Y. Xiang // Physics. Letters A, 2019. – V. 383. – Iss. 24. – P. 2963 – 2968.
4. Бубликов, К. В. Влияние металла на характеристики поперечных мод гибридных волн в слоистой структуре феррит-сегнетоэлектрик / К. В. Бубликов, А. В. Садовников, Е. Н. Бегинин, Ю. П. Шараевский, С. А. Никитов // Письма в журнал технической физики. – 2016. – Т. 42. – Вып. 12. – С. 88–96.
5. Федорова, В. Ю. Функция распределения электронной плотности для металлических наночастиц в рамках теории функционала плотности / В. Ю. Федорова // Политехнический молодежный журнал. – 2018. – № 8. – С. 1–8.
6. Панков, И. А. Распределение поляризации в сферической сегнетоэлектрической наночастице / И. А. Панков, Д. Р. Юнда, А. В. Шуба // Математика, информационные технологии, приложения: сб. тр. межвуз. науч. конф. молодых ученых и студентов (Воронеж, 27 апреля 2022 г.). – Воронеж, 2022. – С. 131–136.
7. Ландау, Л. Д. Теоретическая физика. Т.8. Электродинамика сплошных сред / Л. Д. Ландау, Е. М. Лифшиц. – Москва: Физматлит, 2005. – 656 с.
8. Бейтмен, Г. Высшие трансцендентные функции. Т. 2. Функции Бесселя, функции параболического цилиндра, ортогональные многочлены / Г. Бейтмен, А. Эрдейи. – Москва: Наука, 1966. – 297 с.
9. Цедрик, М. С. Физические свойства кристаллов семейства триглицинсульфата (в зависимости от условий выращивания) / М. С. Цедрик. – Минск: Наука и техника, 1986. – 216 с.
10. Liu, J. Insight into dielectric response of ferroelectric relaxors by statistical modeling / J. Liu, F. Li, Y. Zeng, Z. Jiang, D. Wang, Z.-G. Ye, and C.-L. Jia // Physical Review B. – 2017. – V. 96. – Art. No. 054115 – 11 p.
11. Glinchuk, M. D. The anomalies of the properties of nanomaterials related to the distribution of the grain sizes / M. D. Glinchuk and P. I. Bykov // Condensed Matter. – 2004. – № 6. – Art. No. 0406032 – 9 p.
12. Исупов, В. А. Природа физических явлений в сегнеторелаксорах / В. А. Исупов // Физика твердого тела. – 2003. – Т. 45. – Вып. 6. – С. 1056–1060.
13. Li, S. Diffuse phase transition in ferroelectrics with mesoscopic heterogeneity: Mean-field theory / S. Li, J.A. Eastman, R.E Newham., and L.E. Cross // Physical Review B. – 1997. – V. 55. – № 18. – P. 12067–12077.
14. Кубо, Р. Статистическая механика / Р. Кубо. – Москва: Мир, 1967. – 432 с.

УДК 004

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ СОСТАВЛЕНИЯ СКАЗОК

Д. А. Панкова

Воронежский государственный университет

Введение

У современных детей перед их сверстниками прошлых лет есть определенные преимущества. Они обладают очевидной технической сноровкой, понимают язык современных технологий и быстро адаптируются к изменяющимся условиям. Они намного раньше знакомятся с основами грамоты, чтением, письмом. С самого рождения детей окружают различные электронные устройства и игрушки, и она начинают пользоваться ими также легко, как в свое время нынешние взрослые учились пользоваться техникой. Дети овладевают своеобразным языком, на котором техника «общается» с нами естественно и одновременно со всей лексикой современного им языка.

Важным условием для формирования познавательного интереса у дошкольников является создание насыщенной развивающей среды. Этого можно добиться с использованием информационно-коммуникационных технологий (ИКТ). Применение ИКТ открывает ряд новых возможностей, среди которых игровые и обучающие программы для детей, возможность использования графики и мультимедиа. Использование ИКТ способствует увеличению познавательного интереса, а вместе с ним и общего уровня познавательных возможностей. Одновременное использование текстовой, графической и звуковой информации привлекательно для детей. Особый интерес и вовлеченность в процесс у дошкольников вызывают яркость, динамичность, возможность непосредственно участвовать в происходящем. На развитие ребенка положительно влияет и возможность моделировать разные сюжеты и ситуации, которую предоставляют ИКТ. Немаловажной является игровая составляющая, которая также повышает познавательный интерес. Использование компьютерных технологий помогает развивать самостоятельность, внимание и усидчивость [1].

В воспитании ребенка неотъемлемым элементом всегда были и остаются сказки. На понятном для детей языке они рассказывают о добре и зле, учат жизни. Через образы сказочных героев ребенок учится понимать их внутренний мир, сопереживать им. Помимо этого сказки помогают расширять словарный запас и связную логическую речи. Сказки развивают важнейшие коммуникативные и интеллектуальные навыки, такие как образное мышление, внимание, фантазию, все виды памяти [2]. Сказка дает ребенку возможность сопоставлять, анализировать, делать выводы, погружаясь в красочный мир персонажей.

Самый продуктивный метод обучения и развития ребенка — это игра. Сегодня мы используем различные приложения во всех сферах нашей жизни. Дети уже до трех лет не испытывают трудностей в использовании техники. Институт современных медиа MOMRI заявляет, что игры и развивающие приложения, особенно интерактивные, полезны для детей, а сами дети сегодня являются самой медийной частью современного общества [3].

Однако, какой бы интересной ни была игра на компьютере или планшете, маленькие пользователи не смогут увлечься ей, если внешний вид игры не будет привлекать и удерживать внимание.

1. Особенности приложения

Целью работы было создание приложения, с помощью которого можно будет составлять сказки, выбирая персонажей и места действия, где будет проходить действие сказки.

Так как приложение ориентировано, в первую очередь, на дошкольников, то его интерфейс должен быть максимально привлекательным для этой аудитории. Исследования доказывают, что дети и взрослые воспринимают информацию по-разному. Следовательно, ориентированное на взрослого человека приложение не подойдет ребенку. Также необходимо учесть особенности детских книг. Характерными чертами детской литературы являются соответствующая тематика, преобладание иллюстраций над текстом, простой язык, небольшое количество описаний. Маленькие дети больше рассматривают книгу, как красивую и интересную игрушку, а не читают ее, причем каждая страница книги имеет самостоятельное значение [4]. Поэтому все иллюстрации для своего приложения я разработала самостоятельно, выбрав подходящую стилистику, цветовую гамму, изучив особенности детской иллюстрации.

В начале работы был проанализирован рынок веб-приложений, ориентированных на детскую аудиторию. Выяснилось, что на русском языке подобных решений не представлено, что подтверждает актуальность создания приложения.

Реализация в виде веб-приложения была выбрана потому, что такие приложения являются универсальными, гибкими и удобными в использовании. Они имеют много преимуществ перед системами, работающими по технологии клиент-сервер. Веб-приложение достаточно разместить на хостинге и работать можно с любого устройства, имеющего доступ к интернету. Кроме того, веб-приложение просто поддерживать и администрировать: нет необходимости устанавливать приложение на каждое устройство, обновление происходит при загрузке страницы.

Предлагаемая функциональность приложения:

- возможность составлять сказки с использованием предоставляемых инструментов;
- возможность добавлять, удалять и редактировать отдельные части сказки;
- возможность сохранить составленную сказку на устройство пользователя.

Типичный сценарий использования приложения выглядит следующим образом:

- Пользователь выбирает в главном меню пункт «составить сказку»;
- пользователь производит выбор персонажей, локаций, выбирает порядок следования локаций друг за другом;
- пользователь нажимает кнопку «составить», после чего загружается страница с готовой сказкой, которую при желании можно загрузить в формате PDF на устройство пользователя.

На рис. 1 представлена компонентная схема разработанного приложения.

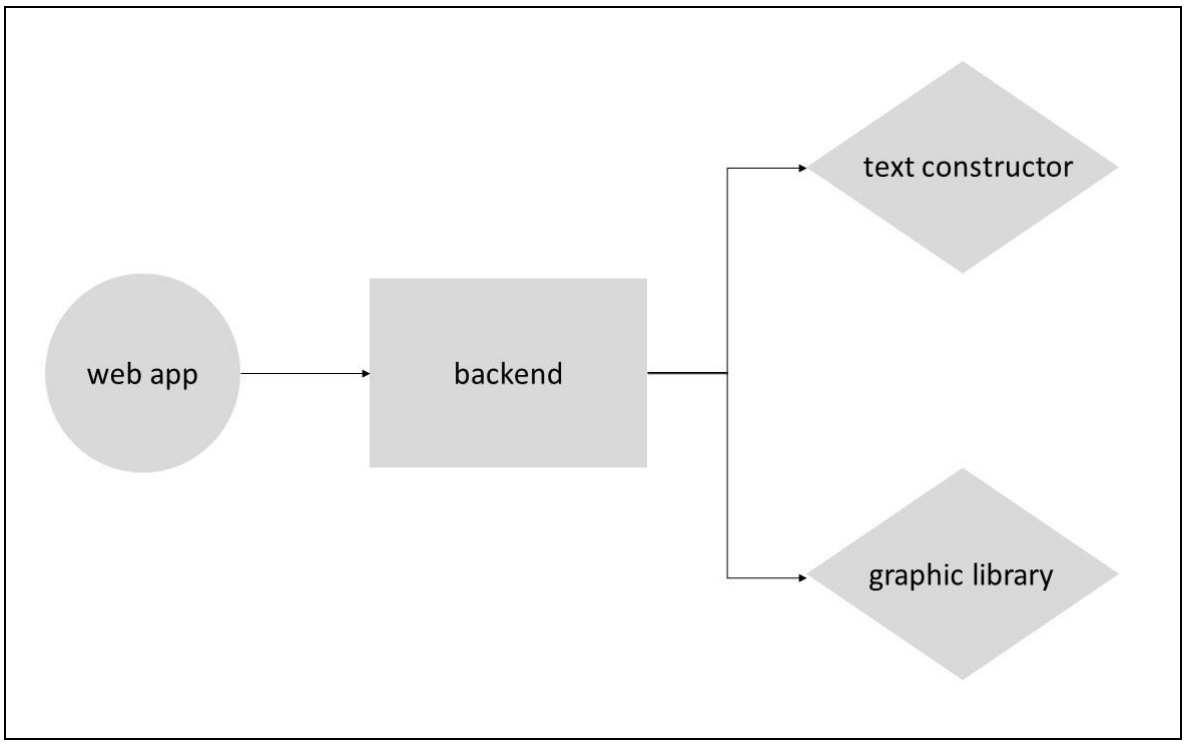


Рис. 1. Компонентная схема приложения

2. Реализация

Для визуализации спроектированной модели данных приложения был использован сервис проектирования моделей баз данных Erwin DataModeler. На рис. 2 представлена логическая модель данных на уровне сущностей.

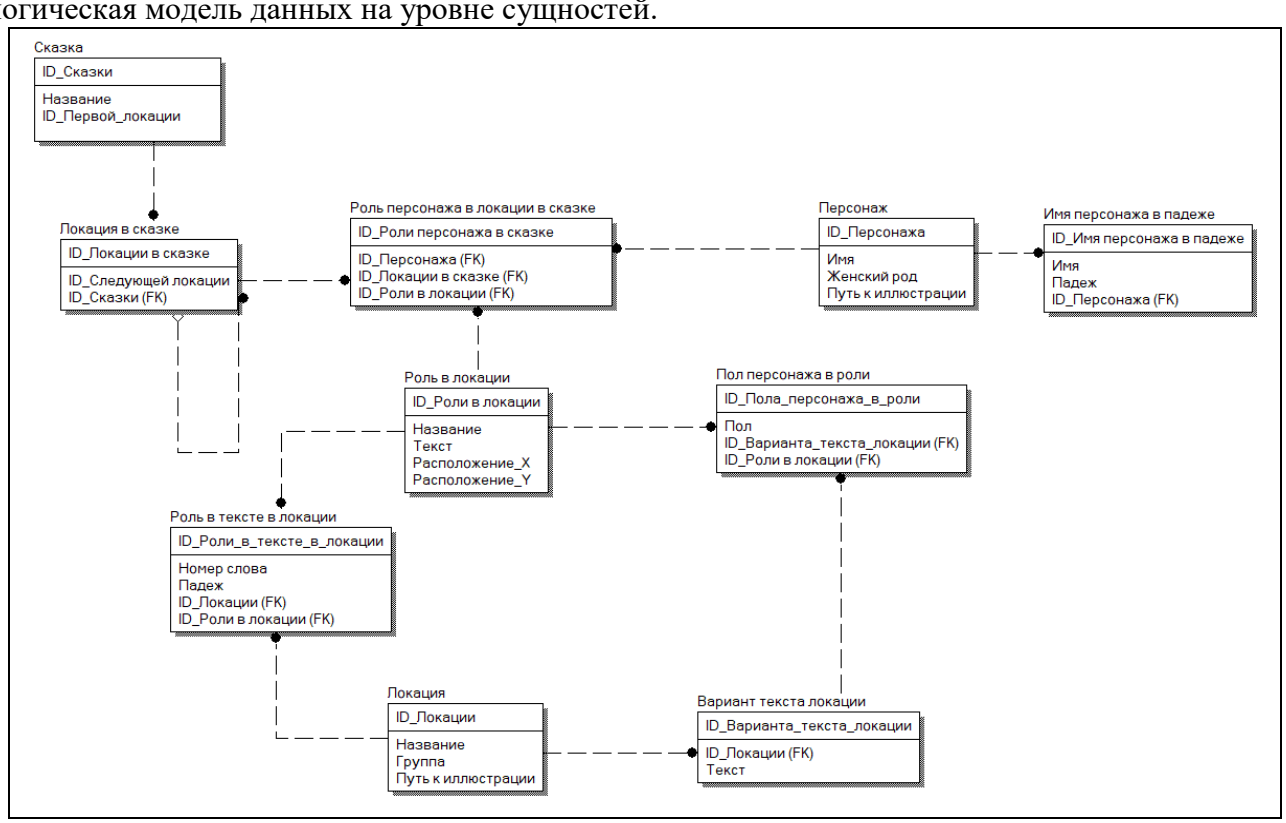


Рис. 2. Логическая модель данных

На рис. 3 представлена функциональная модель сценария использования приложения, построенная в графической модели IDEF0 с помощью CASE-средства Allfusion Erwin Process Modeler.

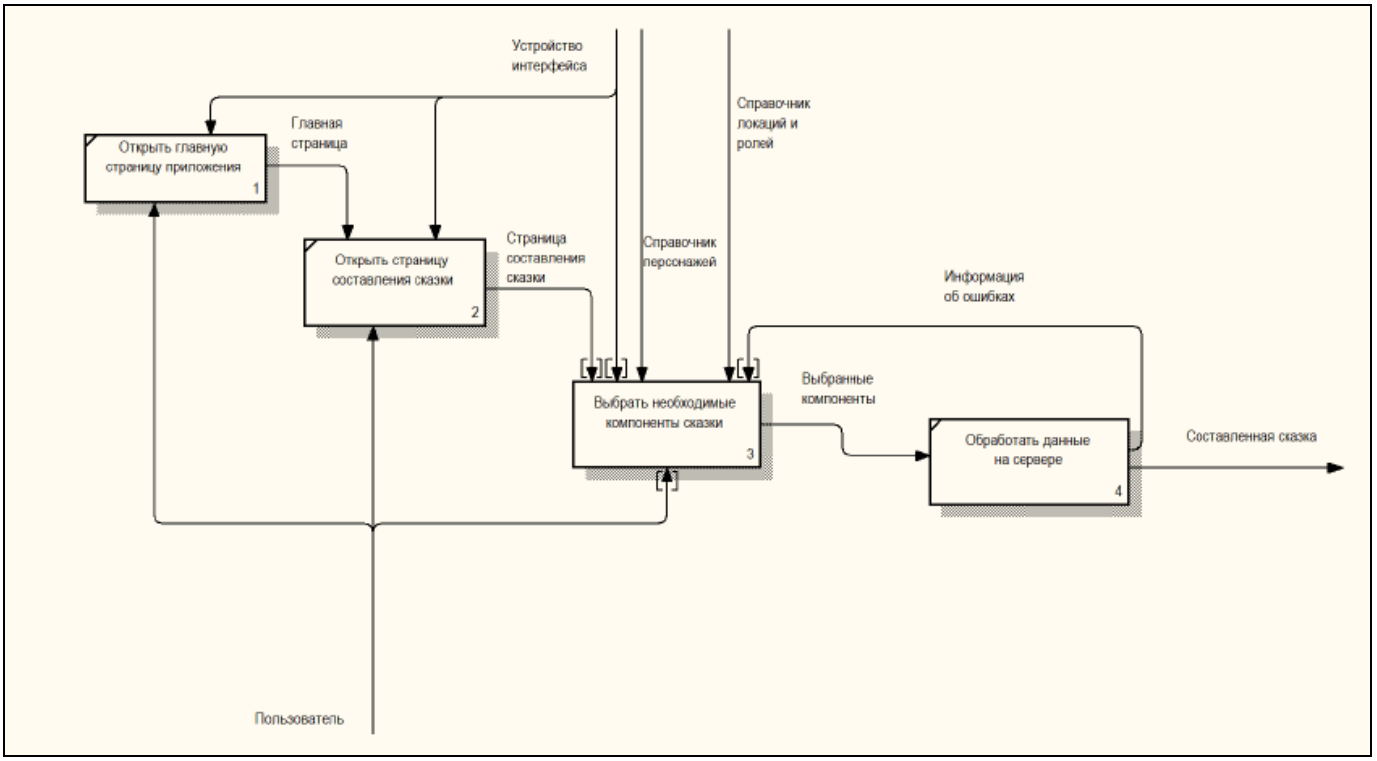


Рис. 3. Функциональная модель сценария использования приложения

При разработке приложений выбор средств реализации представляет сложную задачу и является очень важным этапом. Программные продукты должны удовлетворять не только текущим, но и будущим потребностям. Были выбраны следующие средства:

- язык разработки серверной части приложения — C#;
- среда разработки — Microsoft Visual Studio 2019;
- система контроля версий — Git;
- фреймворк для создания быстродействующих сайтов — Bootstrap;
- фреймворк для создания веб-приложений на платформе .NET — ASP.NET Core;
- ORM-технология Entity Framework Core;
- СУБД — MS SQL Server.

Заключение

По утверждению немецкого психолога Эдик Эриксона, личность человека начинает формироваться уже с первых дней жизни. В первый год дети начинают активно изучать мир доступными методами: пробовать предметы на вкус, трогать, разбросать или сломать. Так проявляется тяга к познанию.

Чуть позже с понятия «кто я такой» начинается личностное развитие. В речи появляется местоимение «Я», дети воспринимают все игрушки как свои.

В возрасте 3-4 лет ребенок строит простые предложения и может выражать свои мысли, постепенно овладевая грамматической структурой речи. В то же время он усваивает навыки общения, в том числе в процессе игры. Таким образом, у детей начинают развиваться познавательные навыки, в приобретении которых важную роль играют различные дидактические игры, а также собственные чувства.

В 4-5 лет развивается образное мышление, преобладает игровая деятельность, в процессе которой ребенок старается понять окружающий его мир [5].

Приложение «Сказка» будет способствовать развитию и обучению дошкольников, совмещая игру и чтение книг.

Литература

1. Москвина А. С. Изобразительное искусство и литературное медиатворчество в организации трудового воспитания детей дошкольного и школьного возрастов / А. С. Москвина, О. И. Радомская, О. В. Юдушкина // Педагогика искусства. – 2016. – № 3. – С. 66–73.

2. Роль сказки в жизни ребенка. – Режим доступа: http://gdoutcrrds32ofprkovvvaar.voadm.gov.spb.ru/_tbkp/Cpiridonova/rol_skazki_v_zhizni_rebenka.pdf. – (Дата обращения: 27.03.2023).

3. Мобильные игры и развивающие приложения полезны для детей. – Режим доступа: https://mospravda.ru/2016/12/09/20855/. – (Дата обращения: 29.03.2023).

4. Макарова К. В. Особенности детской книжной иллюстрации и ее отличия от взрослой / К. В. Макарова // Преподаватель XXI век. – 2010. – Т. 1, № 1. – С. 143–145.

5. Мусаева С. Ш. Интеллектуальное развитие детей 3-4 лет / С. Ш. Мусаева // International scientific review. – 2020. – № 67. – С. 63–65.

УДК: 004.42

АВТОМАТИЗИРОВАННЫЙ ЗАПУСК МАРКЕТИНГОВЫХ КАМПАНИЙ СОГЛАСНО ЗАДАННОМУ РАСПИСАНИЮ С ИСПОЛЬЗОВАНИЕМ ПЛАНИРОВЩИКА ЗАДАНИЙ QUARTZ

А. С. Панфилова

Воронежский государственный университет

Введение

Сложность взаимодействия с клиентом в условиях, когда много отвлекающих и переключающих внимание факторов, вынуждает строить достаточно сложные концепции сценариев взаимодействия с клиентом, в рамках которых учитываются предпочтения, цели, мотивы и потребности каждого из них, что позволяет определить совокупность действий, необходимых для применения к потенциальному покупателю с целью приближения к продукту. Различные сценарии взаимодействия реализуются в рамках отдельных маркетинговых кампаний.

Поскольку данные о клиентах постоянно изменяются, существует необходимость в реализации регламентных запусков маркетинговых кампаний для выделения наиболее релевантного клиентского сегмента для взаимодействия.

1. Анализ логики работы маркетинговой кампании

В общем случае работа маркетинговой кампании состоит из следующих этапов:

1. Запуск расчёта целевого сегмента. Т.е. формирование списка клиентов, по которым необходимо запустить указанную кампанию. Например, необходимо отбирать клиентов, у которых сегодня день рождения.

2. Проверка политики коммуникаций. На данном этапе из целевого сегмента согласно заранее настроенной политике исключаются те клиенты, которые были ранее проинформированы. Данный этап является опциональным и в ряде сценариев может быть опущен.

3. Распределение отобранных клиентов в указанную маркетинговую кампанию. На данном этапе происходит фиксация участников кампании для проведения аналитики и учёта политики коммуникаций в рамках других кампаний.

4. Выполнение определенного действия, настроенного в рамках сценария работы кампании. На этом этапе уже осуществляется непосредственно манипуляция с участником кампании, например, информирование. В случае, если в рамках кампании настроены многошаговые сценарии (например, информирование и начисление бонуса), то процесс выполнения действий будет осуществляться поэтапно: после выполнения очередного действия бу-дут найдены действия, связанные с выполненным. Таким образом, рекурсивно, будет осуществляться обработка всей цепочки действий.

В случае, если кампания имеет регулярный запуск (ежедневно, еженедельно, ежемесячно), её запуск будет осуществляться аналогично п.1–4 с учётом заданного расписания.

Общая схема работы маркетинговой кампании представлена на рис.1.

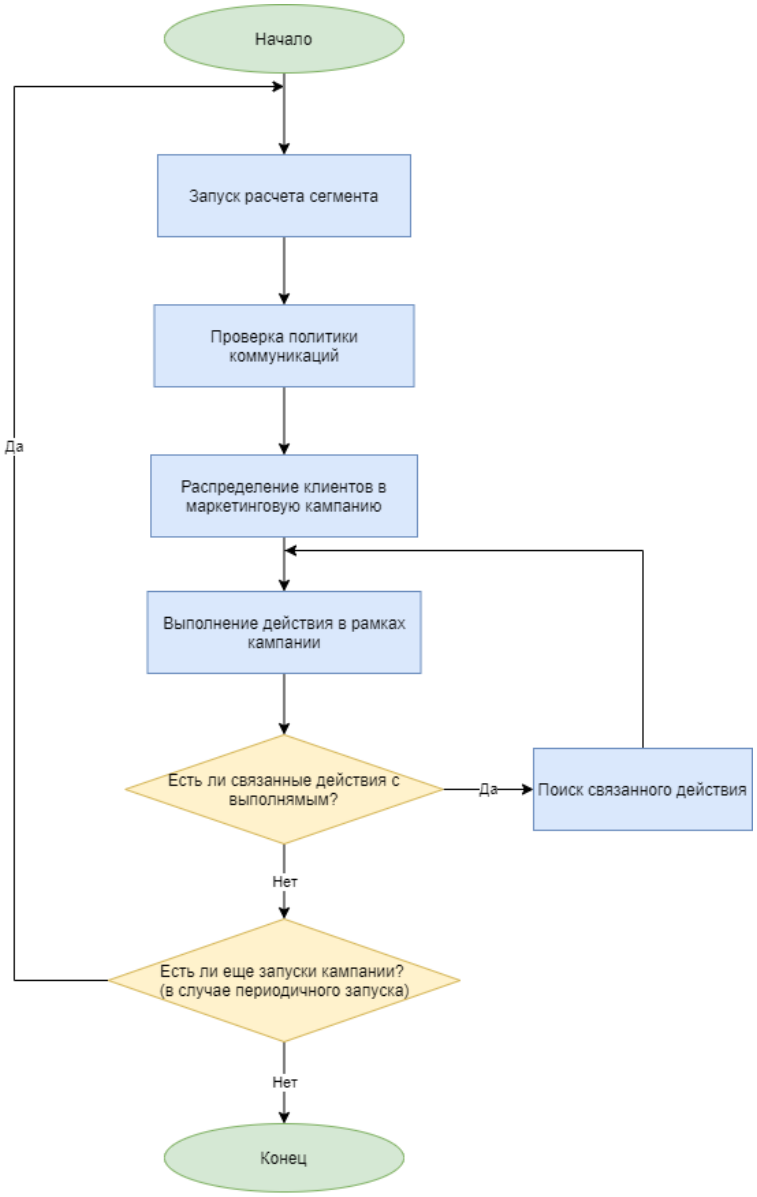


Рис. 1. Общая схема работы маркетинговой кампании

2. Постановка задачи

Необходимо реализовать возможность автоматического запуска маркетинговых кампаний в ручном режиме или согласно заданному расписанию.

Следует иметь возможность задавать следующую периодичность запусков:

- ежедневно (в определенный дни недели);
- еженедельно;
- ежемесячно (в определенный день месяца);
- ежегодно;
- разовый запуск в текущий момент времени;
- разовый отложенный запуск в определенную дату;

При этом срок окончания действия кампании может определяться несколькими способами:

- окончание по наступлению определенной даты;
- окончание после определенного числа запусков;

- без окончания, кампания работает бесконечно.

3. Реализация

3.1. Средства реализации

Для решения поставленной задачи выбраны следующие программные средства:

- среда разработки IntelliJ IDEA;
- язык программирования Java [1];
- библиотека для выполнения запланированных заданий Quartz [4, 5, 6];
- СУБД PostgreSQL [2, 3];
- язык разметки HTML 5.2;

язык программирования JavaScript.

3.2. Общее описание реализации

Для решение поставленной задачи по составлению расписания запусков кампании было принято решение использовать cron-выражения.

Cron-выражение – это строка для задания расписания, состоящая из 6 или 7 полей, разделенных пробелом. Описание полей cron-выражения представлено в табл. 1.

Таблица 1

Описание полей cron-выражения

Описание поля	Необходимость для заполнения	Допустимые значения	Возможные специальные символы
Секунды	+	0-59	, - * /
Минуты	+	0-59	, - * /
Часы	+	0-23	, - * /
День месяца	+	1-31	, - * ? / L W
Месяц	+	1-12 или JAN-DEC	, - * /
День недели	+	1-7 или SUN-SAT	, - * ? / L
Год	-	empty, 1970-2099	, - * /

Возможные специальные символы:

- * (каждое значение) – используется для выбора всех значений в поле;
- ? (без конкретного значения) – используется в случае, когда конкретное значение в поле не принципиально;
- - – используется для указания диапазонов;
- , – используется для указания перечисления значений;
- / – используется для указания приращений;
- L (последний) – последнее из допустимых значений поля;
- W (будний день) – используется для указания буднего дня недели (понедельник-пятница);

Пример cron-выражения для задания условия расписания «каждый будний день в 10 часов 15 минут»: 0 15 10 ? * MON-FRI.

Общая схема работы представлена на рис. 2.

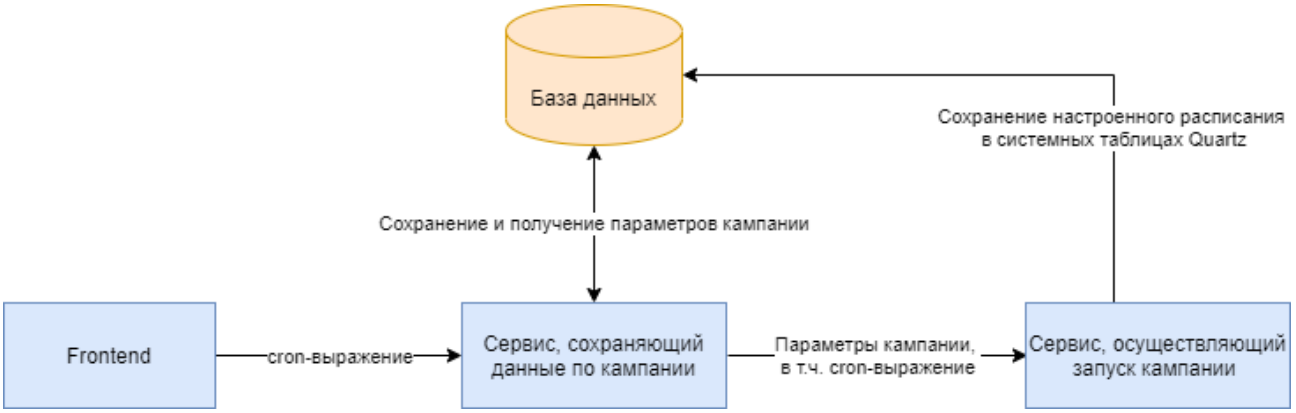


Рис. 2. Общая схема работы процесса запуска кампании

Согласно выбранным пользователем условиям в frontend части приложения формируется cron-выражание, которое сохраняется сервисом как отдельный параметр кампании. Также опционально в рамках отдельного параметра может передаваться и сохраняться дата окончания действия кампании и число запусков.

3.3. Особенности библиотеки Quartz

Идея библиотеки Quartz заключается в том, что планировщик в некотором хранилище данных содержит список заданий, которые запускаются в определённое время или неоднократно.

Основные компоненты Quartz:

- Job (задание) – представляет собой набор операций, которые необходимо выполнить согласно заданному расписанию. Заданию может быть присвоен уникальный идентификатор.
- Trigger – компонент, определяющий расписание, по которому будет выполняться задание. Несколько триггеров могут ссылаться на одно задание.
- Scheduler (планировщик) – отвечает за выполнение заданий, когда срабатывают соответствующие триггеры.

Виды триггеров:

- простые триггеры – используются, когда задание необходимо выполнить ровно один раз в определённый момент времени или в определенный момент времени с последующим повторением через определённый интервал.
- триггеры, работающие на основе cron-выражений – используются, когда необходим график запуска заданий, который повторяется на основе календарных понятий, а не на точно определённых интервалах.

Общая схема работы Quartz представлена на рис. 3.

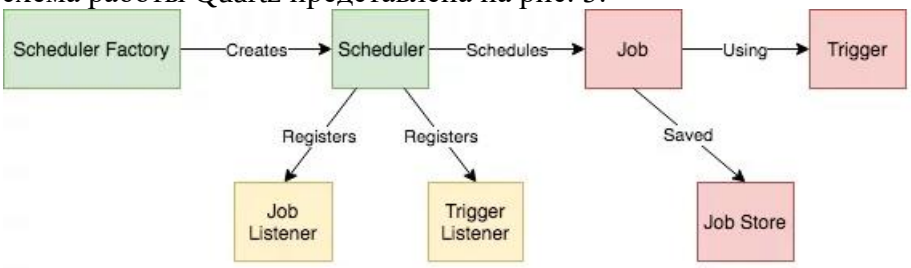


Рис. 3. Общая схема работы Quartz

Для хранения мета-данных о триггерах и заданиях используются следующие таблицы в

реляционной СУБД:

- triggers – содержит общую информацию обо всех триггерах;
- cron_triggers – содержит информацию о триггерах, созданных на основе cron-выражений;
- simple_triggers – содержит информацию о простых триггерах:
- job_details – содержит информацию о выполняемых заданиях;
- fired_triggers – содержит информацию обо всех сработавших триггерах;
- paused_trigger_grps – предназначена для сохранения информации обо всех неактивных триггерах;
- scheduler_state – содержит информацию о состоянии работы экземпляра с целью реализации отказоустойчивости;
- locks – предназначена для хранения значения имени экземпляра, выполняющего задание, для предотвращения ситуации при которой, несколько экземпляров запустят одновременно одно и то же задание.

Модель данных представлена на рис. 4.

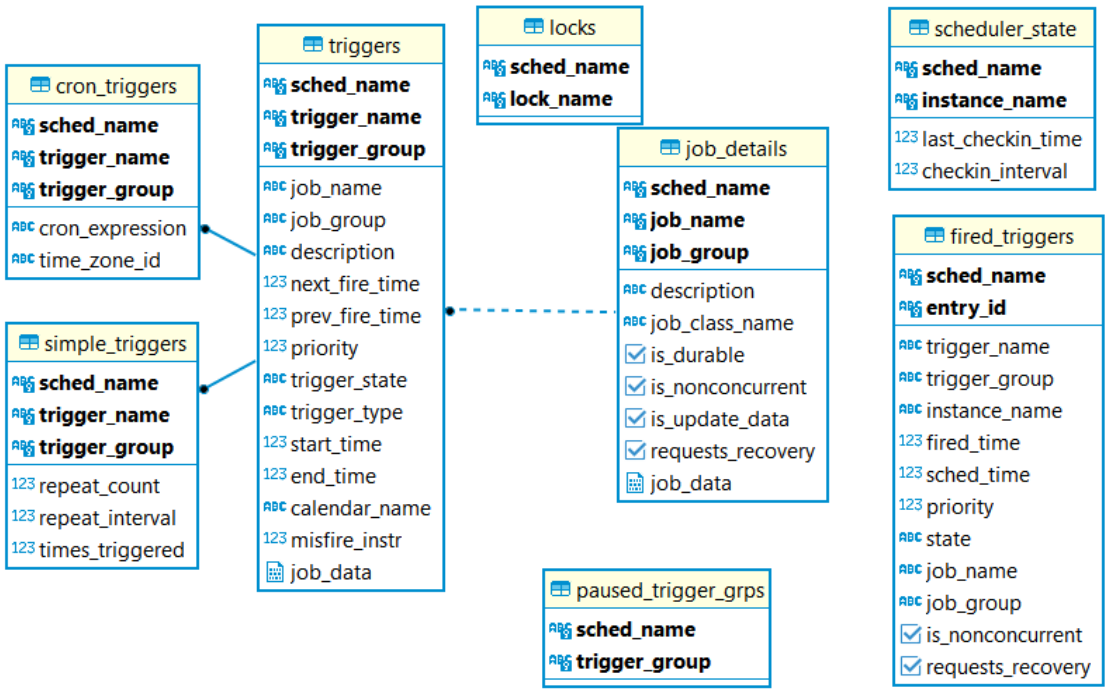


Рис. 4. Модель данных, используемая в библиотеке Quartz

В рамках решения поставленной задачи для каждой кампании создаётся соответствующий триггер, который инициирует вызов метода запуска кампании согласно заданному пользователем расписанию. При остановке кампании пользователем соответствующий триггер становится неактивным.

4. Интерфейс пользователя

Интерфейс пользователя для задания расписания работы маркетинговой кампании представлен на рис. 5–6.

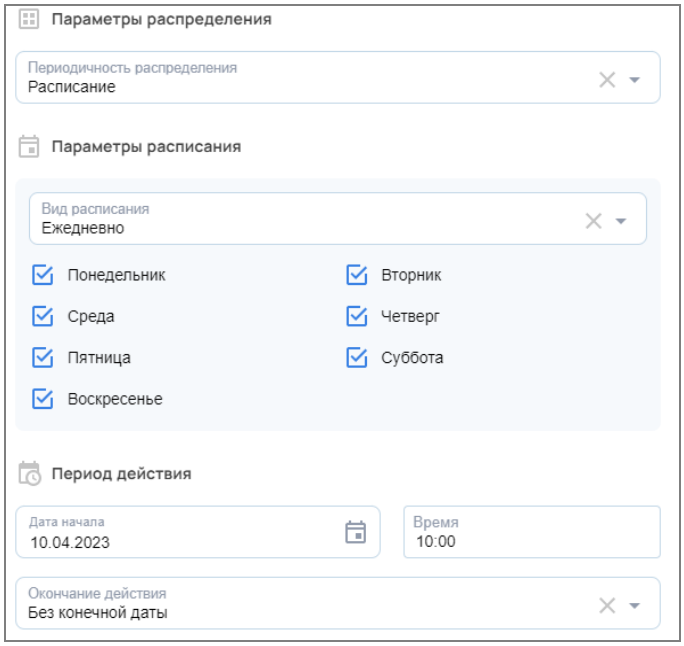


Рис. 5. Интерфейс пользователя для задания ежедневного расписания запусков кампании

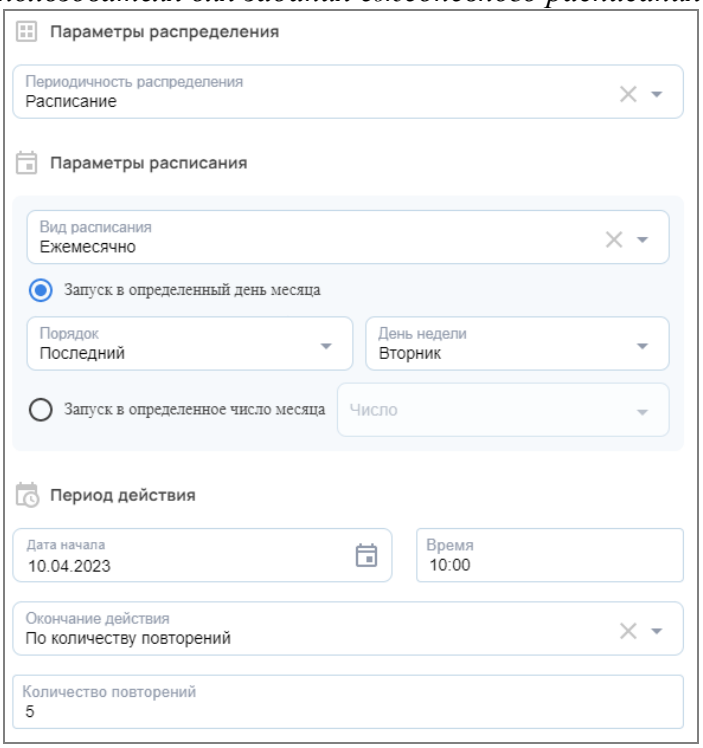


Рис. 6. Интерфейс пользователя для задания ежемесячного расписания запусков кампании

Заключение

Результатом работы является модуль, реализующий возможность автоматического запуска маркетинговых кампаний в ручном режиме или согласно заданному расписанию с использованием библиотеки Quartz.

Научный руководитель: доцент, канд. физ.-мат. наук, доцент кафедры программного обеспечения и администрирования информационных систем ВГУ, Барановский Евгений Сергеевич.

Литература

1. *Шилдт Г.* Java 8. Руководство для начинающих / Г. Шилдт – 6-е изд. – Москва : Вильямс, 2016. – 866 с.

2. *Грофф Джеймс Р.* SQL. Полное руководство / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. – Москва : Вильямс, 2014. – 960 с.

3. *Грабер Мартин.* SQL для простых смертных / Мартин Грабер. – Москва : ЛОРИ, 2014. – 378 с.

4. Документация Quartz. – URL: http://www.quartz-scheduler.org/ (дата обращения 13.02.2023).

5. Добавление Quartz в Spring Boot. – URL: https://habr.com/ru/companies/otus/articles/475996/ (дата обращения 18.02.2023).

6. Java Quartz Architecture Example. – URL: https://examples.javacodegeeks.com/java-development/enterprise-java/quartz/java-quartz-architecture-example/ (дата обращения 04.03.2023).

УДК 004

ОПТИМИЗАЦИЯ ТУРИСТИЧЕСКИХ МАРШРУТОВ НА ОСНОВЕ УЛУЧШЕННОГО АЛГОРИТМА ПРЕПОДАВАНИЯ И ОБУЧЕНИЯ

А. А. Переславцева

Воронежский государственный университет

Введение

Индустрия туризма быстро росла на протяжении многих лет, и с развитием технологий значительно увеличилось количество туристов, посещающих свои любимые места. Однако проблема оптимизации маршрута для туристов остается актуальной.

Туризм является основным источником дохода для многих стран, и количество туристов, посещающих популярные направления, быстро растет. Однако планирование маршрута для посещения различных туристических направлений может быть сложной задачей для туристов, особенно когда они ограничены во времени и ресурсах. Под туристическим маршрутом понимается маршрут с определенными характеристиками, который разумно проходит через ряд туристических мест или городов в определенном регионе, чтобы посетители могли получить максимальный эффект от просмотра в кратчайшие сроки.

Проблема оптимизации маршрута для туристов широко изучалась в области компьютерных наук. Для решения этой задачи было предложено множество алгоритмов. С развитием компьютерных технологий и географических информационных технологий появились такие теории, как ГИС-технологии, технологии интеллектуального анализа данных. Алгоритмы оптимизации и математика постепенно стали применяться при проектировании туристических маршрутов.

В настоящей статье представлен усовершенствованный алгоритм оптимизации на основе преподавания и обучения для изучения оптимизации туристических маршрутов с точки зрения туристического спроса. Данный алгоритм предоставляет инструменты для выбора туристами маршрутов путешествий, разработки маршрутов туристических агентств и планирования регионального туризма.

1. Описание задачи оптимизации туристического маршрута

Разработка туристических маршрутов высокого уровня является важной мерой повышения привлекательности туризма. Как правило, дизайн туристических маршрутов редко изучается с точки зрения «потребностей» туристов в туристических продуктах. В мире туристы всегда стараются получить максимальную выгоду при наименьших затратах как времени, так и денег.

В этом исследовании время посещения туристических достопримечательностей относительно постоянно. Для туристов, помимо выбранных туристических достопримечательностей, расстояние и время являются первым фактором, который они учитывают. Таким образом, в процессе оптимизации туристических маршрутов время в пути и расстояние между живописными местами являются ключевыми факторами, которые необходимо брать во внимание, и которые также станут ограничениями при построении модели.

2. Построение модели оптимизации туристического маршрута

Когда туристы выбирают туристические направления или живописные места, они всегда надеятся удовлетворить свои собственные потребности. Введем $V = \{v_1, v_2 \ldots, v_N\}$— множество, которое представляет собой совокупность всех альтернативных туристических достопримечательностей, где N представляет количество альтернативных туристических достопримечательностей; Из-за ограниченности времени, денег или энергии туристы всегда будут выбирать самые желанные, но при этом доступные живописные места из множества альтернативных живописных мест, тогда $selV = \{sv_1, sv_2 \ldots, sv_n\}$представляет собой набор выбранных туристами живописных мест, где sv_i представляет конкретные живописные места, выбранные туристами, а n ($n \le N$) представляет количество выбранных живописных мест. После выбора пункта назначения можно разработать лучший маршрут тура, используя $V = \{lv_1, lv_2 \ldots, lv_n\}$, где lv_i представляет оптимизированное расположение выбранного набора *selV*.

Туристы могут свободно выбирать начальную точку lv_0и конечную точку lv_{n+1} маршрута тура (в соответствие с рисунком 1).

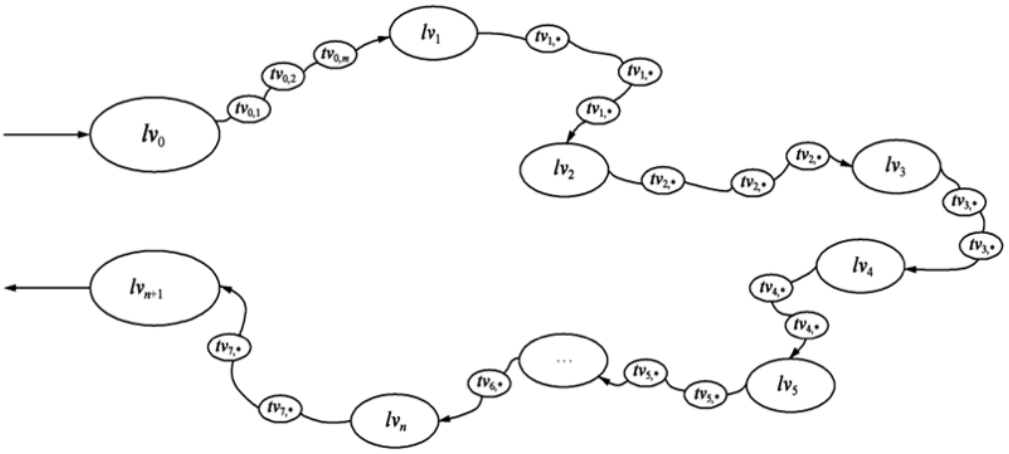


Рис. 1.Карта туристического маршрута

В соответствии с принципом наименьшего времени строится модель минимизации времени туристского маршрута:

$$minf(t) = \sum_{i=1}^{n} t_1 + \sum_{j=1}^{n} \frac{Distance\left(lv_j, lv_{j+1}\right)}{V\left(lv_j, lv_{j+1}\right)} \tag{1}$$

где

$t_i (i = 1, 2 \ldots, n)$ — время пребывания туристов в живописных местах;

n — количество живописных мест для посещения;

$Distance(lv_j, lv_{j+1}), (j=0,1,2\ldots,n)$ — расстояние от живописного места lv_j до живописного места lv_{j+1};

$V(lv_j, lv_{j+1})$ — скорость движения от живописного места lv_j в живописное место lv_{j+1}.

3. Идеи и методы исследования

3.1.Основные принципы алгоритма на основе преподавания-обучения

Алгоритм на основе преподавания-обучения (TLBO) [1] — это новый метаэвристический алгоритм, который повышает уровень знаний, имитируя «Преподавание» и «Обучение» в процессе обучения людей. TLBO имеет несколько параметров и высокую производительность. TLBO был предложен в 2011 году и хорошо применялся для оптимизации механической конструкции, теплообменников, термоэлектрических охладителей и других областях.

Чтобы облегчить понимание, ниже приведены некоторые основные определения алгоритма TLBO:

1. Пространство поиска индивидуума (вектора решения) $X=(x_1, x_2\ldots, x_D)$ называется Ученик, $x_i(i=1,2\ldots,D)$ — i-й курс для студентов.
2. Множество всех учащихся называется классом.
3. Учащиеся с самым высоким уровнем (подготовленности) $X^{best}=(x_1^{best}, x_2^{best}\ldots, x_D^{best})$ называются Учителями.

В алгоритме TLBO класс равен населению в генетическом алгоритме, ученик равен индивидууму, а учитель — индивидууму с наивысшей адаптивной ценностью. Задача учителя — усердно учить и продвигать средний уровень учащихся в классе. Студенты совершенствуют свои знания, учась у учителей и общаясь с одноклассниками. Алгоритм TLBO разделен на два этапа: этап преподавания и этап обучения [2] .

1) Алгоритм этапа преподавания указан на рисунке 2:

$$\text{For each learner } X^j = (x_1^j, x_2^j, \ldots, x_D^j)(j = 1, 2, \ldots, NP) \text{ Do}$$

$$x_i^{j,new} = x_i^{j,old} + rand() \cdot (x_i^{best} - T_F \cdot Mean_i), j = 1, 2, \ldots, NP, i = 1, 2, .., D$$

$$\text{If } X^{j,new} \text{ first } X^{j,old} \text{ then}$$

$$\text{End if}$$

$$\text{End for}$$

Рис. 2. Алгоритм TLBO. Этап преподавания.

где

$x_i^{j,old}, x_i^{j,new}$ представляют уровень знаний X_j i-го курса до и после обучения соответственно;

Rand() — это случайное число в диапазоне (0,1);

NP — это общее количество студентов;

D — это количество курсов (размерность);

$$T_F = round[1 + rand(\ )], Mean_i = \frac{1}{NP}\sum_{j=1}^{NP} x_i^j$$

2) Алгоритм этапа обучения указан на рисунке 3:

For each learner $X^j(j = 1, 2 \ldots NP)$
Select a student X^k at random from the class $(j \neq k)$
If X^j is superior to X^k then
$X^{j,new} = X^{j,old} + rand(1, D) \cdot (X^j - X^k)$
Else
$X^{j,new} = X^{j,old} + rand(1, D) \cdot (X^k - X^j)$
End
$X^{j,new}$ is superior to $X^{j,old}$ then
$X^j = X^{j,new}$
End if
End for

Рис. 3. Алгоритм TLBO. Этап обучения.

Где r*and(1,D)* означает, что D-мерный вектор генерируется случайным образом в интервале (0,1).

3.2. Оптимизация туристических маршрутов на основе TLBO

Базовый алгоритм TLBO в основном используется для задач оптимизации действительных чисел. Для задач дискретной комбинаторной оптимизации его необходимо переработать. В этой статье представлен усовершенствованный алгоритм преподавания и обучения (ITLBO) для планирования туристических маршрутов. Его методы «преподавания» и «обучения» следующие:

1) Фаза преподавания

На этапе обучения каждый учащийся использует PMX, оператор перехода с частичным соответствием, чтобы учиться у Учителя. Алгоритм следующий:

For each learner $X^j(j = 1, 2 \ldots, NP)$ Do
$X^{new} = X^j$
// Select a random element R_s in the set $\{1, 2, \ldots D - Step\}$
$Re = Rs + Step$ (as a closing crossover point)
// Partially matched crossover (PMX) algorithm
for X^{new} and $X^{teacher}$
If X^{new} is better than X^i
$X^i = X^{new}$
End if
End for

Рис. 4. Алгоритм ITLBO. Этап преподавания.

Среди которых, размер перекрестного шага динамически корректируется по мере продвижения оптимизации, что помогает алгоритму достичь баланса между глобальным исследованием и локальным уточнением.

$$Step = \begin{cases} Int\left(\frac{D}{2}\right) + 1, & t < T_{max}/5, \\ Int\left(\frac{D}{3}\right) + 1, & t < 2T_{max}/5, \\ Int\left(\frac{D}{4}\right) + 1, & t < 3T_{max}/5, \\ Int\left(\frac{D}{5}\right) + 1, & t < 4T_{max}/5, \\ 1, & otherwise \end{cases}$$

2) Фаза обучения

На этапе взаимного обучения учащиеся случайным образом выбирают других учащихся для перекрестного обучения, используя следующие методы, как представлено на рисунке 5:

For each learner $X^i (i = 1, 2 \ldots, NP)$

$X^{new} = X^i$

$//X^j$ students were randomly selected $(j = 1, 2 \ldots NP, j \neq i)$ as the learning object

// Randomly selected crossover starting point $R_s \in \{1, 2 \ldots, D - Step\}$, end point $R_e = R_s + Step$

// Perform partial matching crossover operator (PMX) on X^{new} and X^j

If X^{new} is better than X^i

$X^i = X^{new}$

End if

End for

Рис. 5. Алгоритм ITLBO. Этап обучения.

В базовом алгоритме TLBO уровень знаний учащихся повышается за счет обучения у учителей и общения с одноклассниками. Однако, помимо обучения у преподавателей и студентов самым важным должно быть «самообучение», а процесс самообучения не отражен в базовом алгоритме TLBO.

3) Фаза самообучения

Алгоритм этапа «самообучения» следующий представлен на рисунке 6:

For each learner $X^i(i = 1, 2\ldots, NP)$ Do

$X^{new} = X^i$

// Execute two-point crossover on X^{new} at J and

//J_round (J is {1,2..., a random integer in D}, J_round is a random integer near J)

If X^{new} is better than X^i

$X^i = X^{new}$

End if

End for

Рис. 6. Алгоритм ITLBO. Этап самообучения.

Блок схема всего алгоритма представлена на рисунке 7 [3] :

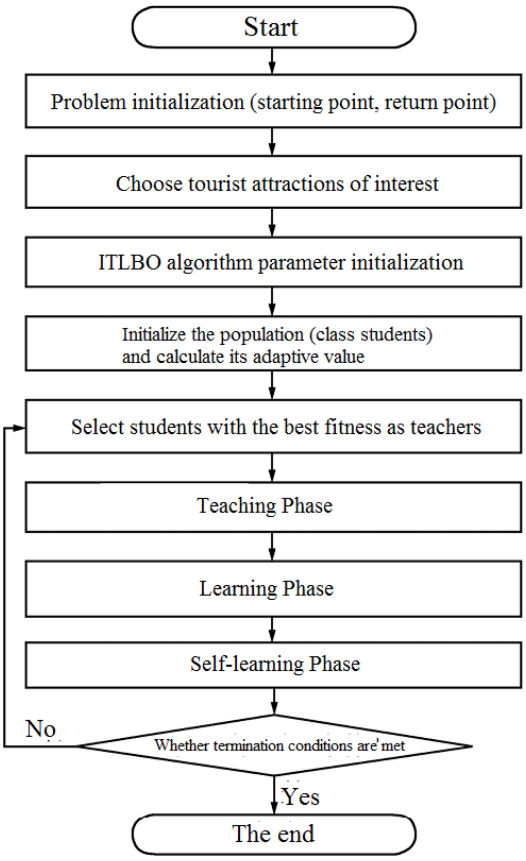


Рис. 7. Блок-схема оптимизации маршрута путешествия по алгоритму ITLBO.

Заключение

В данной статье был рассмотрен алгоритм оптимизации туристического движения, основанный на улучшении алгоритма преподавания и обучения. Эмпирическое исследование, проведенное на основе данного алгоритма, посредством моделирования туристических достопримечательностей, выбора туриста для расчета более идеальных и эффективных маршрутов путешествия, показало улучшенные результаты по сравнению с традиционным

путешествием. Агентства по проектированию схем обнаружили, что алгоритм разработки схемы имеет определенную осуществимость. Он может служить ориентиром для практического планирования туристического движения и разработки маршрута, особенно актуального для туристов с самостоятельным вождением или индивидуальных туристов.

В этом исследовании еще предстоит изучить множество областей. Это исследование опирается на существующее движение по скоростным автомагистралям в качестве схемы маршрута, без учета железнодорожного, водного транспорта и воздушного движения. В данном исследовании считается, что время посещения и время размещения туристов относительно фиксированы, а различные непредвиденные ситуации не рассматриваются. При проектировании маршрута учитывается только принцип кратчайшего времени в пути, а расстояние положительно коррелирует со временем в пути, а проблема стоимости проезда не рассматривается. Это исследование в основном предназначено для самостоятельных туристов, но в нем нет углубленного изучения конкретных предпочтений спроса и особых потребностей самоходных туристов. Таким образом, следующим шагом является учет затрат на поездки, дальнейшее понимание характеристик спроса на самоходных туристов и разработка целевых, научно-обоснованных и разумных маршрутов поездок.

Литература

1. What is Teaching Learning Based Optimization (TLBO) Algorithm. – Режим доступа: https://medium.com/analytics-vidhya/what-is-teaching-learning-based-optimization-tlbo-algorithm-b368098339e8. – (Дата обращения: 18.04.2023).

2. Teaching Learning based Optimization (TLBO). – Режим доступа: https://www.geeksforgeeks.org/teaching-learning-based-optimization-tlbo/. – (Дата обращения: 18.04.2023).

3. Flow chart for Teaching-Learning Based Optimization (TLBO) algorithm. – Режим доступа: https://www.researchgate.net/figure/Flow-chart-for-Teaching-Learning-Based-Optimization-TLBO-algorithm_fig1_283497481. – (Дата обращения: 18.04.2023).

УДК 519.673

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ПРЕСНОВОДНЫХ ГИДР

А.А. Петина

Воронежский государственный университет

Введение

Задачи поиска оптимальных решений часто возникают как в науке, так и в прикладных областях человеческой деятельности. Алгоритмы, относящиеся к классу популяционных, являются эффективным способом оптимизации n-мерных функций, имеющих высокую вычислительную сложность. Одним из таких алгоритмов является алгоритм пресноводных гидр [1].

Не при всех параметрах алгоритм может быть эффективен при решении тех или иных задач, поэтому актуальными являются проблема создания программной системы, предоставляющей пользователю возможность тестирования алгоритма, и проблема решения задачи параметрической метаоптимизации популяционных алгоритмов глобальной оптимизации, суть которой заключается в отыскании оптимальных в некотором смысле значений их свободных параметров [2].

1. Краткое описание алгоритма

В алгоритм заложены механизмы, подобные поведению пресноводных гидр. Так, при наступлении неблагоприятных условий среды, гидра погибает, выпуская в воду яйцо, переносимое с током воды, из которого вырастает новая особь гидры. Этот факт формализован в алгоритме с помощью оператора переноса. Особь гидры может перемещаться к средней лучшей позиции всех особей популяции, к особи с наилучшим значением целевой функции за все время поиска и в случайном направлении. При этом выбор направления выполняется по-разному. В первом варианте алгоритма (QH-AHP-алгоритм) для этого применяется метод анализа иерархий (AHP), а во втором (QH-B-алгоритм) — байесовский подход [1].

2. Модификации алгоритма

Алгоритм работает сразу с несколькими допустимыми векторами, которые называются особями, агентами или частицами. Совокупность особей образует популяцию. Особь с наилучшим значением целевой функции называется лидером [1].

2.1. Модификация на основе метода анализа иерархий (QH-AHP-алгоритм)

Направление движения особи определяется с помощью метода анализа иерархий (Analytic Hierarchy Process, AHP). Этот метод позволяет выбирать решение среди p возможных вариантов. Предпочтения в пользу того или иного варианта выражаются матрицей парных сравнений $\mathbf{A}$. Ее элементы $a_{ij} \in \left\{\frac{1}{10}, \frac{1}{9}, ..., 1, 2, ..., 10\right\}$ показывают, насколько i-й вариант

лучше j-го ($i, j = \overline{1, p}$), причем $a_{ij} = \dfrac{1}{a_{ji}}$ и $a_{ii} = 1$. Нормируя элементы матрицы $\mathbf{A}$ по правилу $b_{ij} = \dfrac{a_{ij}}{\sum_{i=1}^{p} a_{ij}}$, формируется матрица $\mathbf{B}$. Начальные значения элементов матрицы $\mathbf{A}$ представлены в табл. 1.

Таблица 1

Начальные значения элементов матриц парных сравнений

Направление	К средней лучшей позиции	К лидеру	Случайное
К средней лучшей позиции	1	$\frac{1}{3}$	$\frac{1}{7}$
К лидеру	3	1	$\frac{1}{5}$
Случайное	7	5	1

Алгоритм [1]:

Шаг I. Случайным образом создается популяция из S особей x_j ($j = \overline{1, S}$), являющихся возможными решениями. При этом координаты особей x_{ij} ($i = \overline{1, n}$) принимают случайные значения из заданного промежутка $\left[x_i^{\min}, x_i^{\max}\right]$. Для каждой j-й особи задать матрицу парных сравнений $\mathbf{A_j}$ (см. табл. 1), определить значение целевой функции, положить число итераций τ_j, в течение которых целевая функция не улучшалась, равным нулю. Задать номер итерации $k = 1$. Найти лидера.

Шаг II. По формуле

$$C_i^k = \frac{1}{S} \sum_{j=1}^{S} x_{ij}^* \qquad (1)$$

определить средний вектор $\mathbf{C}$ лучших положений особей популяции и найти значение параметра λ с помощью выражения

$$\lambda^k = \lambda_1 + \frac{\lambda_0 - \lambda_1}{k}, \qquad (2)$$

где λ_0 — его начальное значение (при $k = 1$), а к значению λ_1 данный параметр стремится при $k \to \infty$.

Шаг III. Для каждой j-й особи выполнить следующее:

1. Найти вектор приоритетов $\mathbf{w}$, где степень предпочтения i-го варианта выражается его весовым коэффициентом, равным среднему значению элементов i-й строки матрицы $\mathbf{B}$

$$w_i = \frac{1}{p} \sum_{j=1}^{p} b_{ij} . \qquad (3)$$

2. Выполнить шаг в направлении с наибольшим приоритетом $p_{\max} = \arg\max_p w_p$.

- Движение к среднему вектору **C** лучших положений особей популяции выполняется по формулам:

$$x_{ij}^k = \begin{cases} p_{ij}^k + \lambda^k \cdot \left| C_i^k - x_{ij}^k \right| \cdot \ln \dfrac{1}{\alpha_{ij}^k}, \beta_{ij}^k \geq 0,5 \\ p_{ij}^k - \lambda^k \cdot \left| C_i^k - x_{ij}^k \right| \cdot \ln \dfrac{1}{\alpha_{ij}^k}, \beta_{ij}^k < 0,5 \end{cases}, \quad (4)$$

где случайные величины α_{ij} и β_{ij} принадлежат интервалу $(0,1)$.

Координаты притяжения каждой особи определяются следующим образом:

$$p_{ij}^k = \varphi_{ij}^k x_{ij}^* + (1-\varphi_{ij}^k) x_i^{**}, \quad (5)$$

где x_i^{**} — координаты лучшего решения (лидера) за все время поиска; случайные величины φ_{ij} принимают значения из диапазона $[0,1]$.

- Для каждой координаты особи задаются две случайные величины γ_{ij} и χ_{ij}, принадлежащие интервалу $(0,1)$. На их основе вычисляются две независимые случайные величины:

$$\xi_{ij} = \sqrt{-2\ln\gamma_{ij}} \cdot \cos 2\pi\chi_{ij},$$
$$\eta_{ij} = \sqrt{-2\ln\gamma_{ij}} \cdot \sin 2\pi\chi_{ij},$$

закон распределения которых близок к стандартному нормальному $N(0,1)$. Также генерируются равномерно распределенные на промежутке $[0,1]$ случайные величины α_{ij} и β_{ij} .

Переход в направлении к лидеру осуществляется согласно соотношения:

$$x_{ij}^{k+1} = x_{ij}^k + \lambda^k \cdot \xi_{ij}^k \cdot (x_i^{**} - x_{ij}^k) \cdot \alpha_{ij}^k. \quad (6)$$

- Перемещение особи в случайном направлении реализуется следующим образом

$$x_{ij}^k = \begin{cases} x_{ij}^* + \lambda^k \cdot \eta_{ij}^k, \beta_{ij}^k \geq 0,5 \\ x_{ij}^* - \lambda^k \cdot \eta_{ij}^k, \beta_{ij}^k < 0,5 \end{cases}. \quad (7)$$

Если он одинаков у нескольких направлений, то выбрать любое из них.

3. Вычислить значение целевой функции. Если оно улучшилось, то в строке $i = p_{\max}$ матрицы **A** увеличить элементы a_{ij} (например, на 1, если $a_{ij} > 1$, и на 0,1, если $0 < a_{ij} < 1$). Если при этом значение a_{ij} оказывается вне диапазона $[0,1;10]$, то положить его равным ближайшему граничному значению. Положить τ_j равным нулю. Обновить лучшее положение особи $\mathbf{x}_j^*$, если это возможно.

4. Если значение целевой функции ухудшается, то уменьшить соответствующие элементы матрицы **A** . Увеличить τ_j на единицу.

5. Пересчитать элементы, симметричные тем, которые изменились: $a_{ji} = \dfrac{1}{a_{ij}}$.

6. Если $\tau_j = \tau_{\max}$, то применить оператор переноса для увеличения охвата пространства поиска и предотвращения преждевременной сходимости алгоритма к локальному экстремуму:

$$x_{ij}^{k+1} = \begin{cases} x_{ij}^k + \alpha_{ij} \cdot (x_i^{\max} - x_{ij}^k) \cdot g(k), \beta_{ij} \geq 0,5 \\ x_{ij}^k - \alpha_{ij} \cdot (x_i^{\max} - x_{ij}^k) \cdot g(k), \beta_{ij} < 0,5 \end{cases}, \quad (8)$$

где случайные числа α_{ij} и β_{ij} генерируются из промежутка $[0,1]$ для каждой координаты особи x_j^k, а функция $g(k)$ выбирается так, чтобы она принимала значения из промежутка $[0,1]$. В этом случае и значения x_{ij}^{k+1} остаются внутри промежутка $[0,1]$. В данной работе использована функция вида:

$$g(q) = \frac{1}{2} \cdot \left(1 - \frac{q}{\mu + |q|}\right), q = k - \frac{K}{2}.$$

Далее нужно рассчитать значение целевой функции в новой точке и вернуть элементам матрицы $\mathbf{A}_j$ исходные значения (см. табл. 1).

Шаг IV. Найти лидера и проверить выполнение критерия останова:

$$\chi = \max_q \left| \frac{f^k - f^{k-q}}{f^k} \right| < 10^{-4}, \quad (9)$$

где $q = 1, ..., \min(k, \omega)$; значение параметра ω принимается равным 100 итерациям.

В случае выполнения завершить поиск. Положить $k = k + 1$ и перейти к шагу II.

2.2. Модификация на основе байесовского подхода (QH-B-алгоритм)

Для выбора направления движения особи используется байесовский подход. При этом рассматриваются следующие случайные события (гипотезы): H_1 — особь перемещается к среднему вектору $\mathbf{C}$ лучших положений всех особей популяции; H_2 — особь движется в направлении к лидеру; H_3 — перемещение особи осуществляется случайным образом.

В результате перемещения особи в том или ином направлении появляются следующие свидетельства: A — значение целевой функции особи улучшилось; $\overline{A}$ — значение целевой функции особи ухудшилось.

На основе события A делается вывод о правильности той или иной гипотезы. Предполагается, что события H_1, H_2, H_3 образуют полную группу событий. При использовании байесовского подхода исходные априорные вероятности гипотез $P(H_1), P(H_2), P(H_3)$ последовательно пересчитываются на основе поступающих свидетельств A или $\overline{A}$.

Реализация QH-B-алгоритма схожа с реализацией QH-AHP-алгоритма, за исключением следующего [1]:

1) Изначально, вместо матриц парных сравнений, для каждой особи задать вероятности $P(H_1), P(H_2)$, $P(H_3)$.

2) На каждом шаге вычислить вероятности $P(A|H_1), P(A|H_2), P(A|H_3)$, как функции от номера итерации:

$$P(A|H_r) = w_r + \frac{u_r - w_r}{k},$$

где u_r представляет собой начальное значение вероятности ($P(A|H_r) = u_r$ при $k = 1$), а к значению w_r эта вероятность стремится при $k \to \infty$.

3) Если значение целевой функции улучшилось (наступило событие A), то вычислить вероятности $P(H_r)$, считая, что они равны апостериорным вероятностям формулы:

$$P(H_r \mid A) = \frac{P(H_r) \cdot P(A \mid H_r)}{\sum_{s=1}^{3} P(H_s) \cdot P(A \mid H_s)}, \quad (10)$$

где $r \in \{1, 2, 3\}$.

4) Если значение целевой функции ухудшилось, то пересчитать вероятности $P(H_r)$ по формуле:

$$P(H_r \mid \bar{A}) = \frac{P(H_r) \cdot P(\bar{A} \mid H_r)}{\sum_{s=1}^{3} P(H_s) \cdot P(\bar{A} \mid H_s)}, \quad (11)$$

где $r \in \{1, 2, 3\}$.

5) В случае, если $\tau_j = \tau_{\max}$, вернуть вероятностям $P(H_r)$ исходные значения.

3. Вычислительный эксперимент

3.1. Цель вычислительного эксперимента

Целью вычислительного эксперимента является настройка параметров для достижения наиболее эффективной работы алгоритма. Для QH-AHP-алгоритма это параметры: $a_{ij} \in [1,...,10]$, $i < j$, $i, j = 1, 2, 3$, $S \in [5, 53]$, $\lambda_0 \in (0, 10)$, $\lambda_1 \in (0, 10)$, $\mu \in [5, 53]$, $\tau_{\max} \in [1, 30]$. Для QH-B-алгоритма: $P(H_1)$, $P(H_2)$, $P(H_3)$, u_1, u_2, u_3, w_1, w_2, w_3, S, λ_0, λ_1, μ, $\tau_{\max}$. При этом $P(H_r) \in [0, 1]$, $u_r \in [0, 1]$, $w_r \in [0, 1]$, $r = \overline{1, 3}$.

3.2. Описание программы

Для реализации программы необходимо использование объектно-ориентированного программирования. Это обусловлено необходимостью создания классов особи и популяции и многократного применения к ним методов поиска и перемещения [3]. На основе данных факторов был выбран объектно-ориентированный язык C#.

Разработанная программа вычисляет скорость сходимости каждого из приведенных алгоритмов.

В программе используются следующие классы: Population — популяция (содержит в себе массив типа Individual, количество особей в популяции); Individual — особь (содержит текущие координаты особи и лучшие координаты за все время поиска).

К основным методам класса Population относятся следующие:

`void FindLeaderMin()` — поиск лидера в популяции;

`void FindC()` — ищет средний вектор лучших положений по формуле (1);

`void FindThePriorityVector()` — поиск вектора приоритетов по формуле (3);

`void StepInTheHighestPriorityDirection()` — для каждой особи совершает шаг в более приоритетном для нее направлении по формулам (4), (6) или (7).

К основным методам класса Individual относятся следующие:

`int P_max()` — нахождение приоритетного направления для перемещения особи;

`void TransferOperator()` — реализация оператора переноса (8).

3.3. Эксперимент и тестовые функции

Сравнение скорости сходимости рассматриваемых методов выполнялось на примере следующих функций:

a) $f_1(\mathbf{x}) = \sum_{i=1}^{n-1}\left(100\cdot\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right)$ — функция Розенброка. Оптимальное решение $x_i^{opt}=1,\ i=\overline{1,n},\ f_1(\mathbf{x}^{opt})=0.$

b) $f_2(\mathbf{x}) = \sum_{i=1}^{n-1}\left(x_{i+1}^2+x_i^2\right)^{0,25}\times\left(\sin^2\left(50\left(x_{i+1}^2+x_i^2\right)^{0,1}\right)+1\right)$ — функция Дэвиса. Оптимальное решение $x_i^{opt}=0,\ i=\overline{1,n},\ f_2(\mathbf{x}^{opt})=0.$

c) $f_3(\mathbf{x}) = 1+\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)$ — функция Гриванка [4]. Оптимальное решение $x_i^{opt}=0,\ i=\overline{1,n},\ f_3(\mathbf{x}^{opt})=0.$

d) $f_4(\mathbf{x}) = \sum_{i=1}^{n}10\left(1-\cos\left(2\pi x_i\right)\right)+x_i^2$ — функция Растригина. Оптимальное решение $x_i^{opt}=0,\ i=\overline{1,n},\ f_4(\mathbf{x}^{opt})=0.$

В качестве диапазона изменения всех координат x_i был выбран промежуток $[-10,10]$. Рассматривались функции от $n=5$ переменных. Количество особей в популяции принималось равным 50. В роли зависящей от этих параметров целевой метафункции ψ принималось среднее значение целевой функции f_q $\left(q=\overline{1,4}\right)$, достигаемое за 10 запусков алгоритма, причем длительность каждого запуска ограничивалась 100 секундами. Для получения результатов, оптимальных в среднем для всех тестовых функций f_q, на каждой итерации метаалгоритма она выбиралась случайным образом.

3.4. Характеристика полученных данных

На рис. 1 показаны графики зависимости скорости сходимости алгоритмов оптимизации (значение целевое метафункции ψ) от значений настраиваемых параметров алгоритма.

Разработав систему, позволяющую оценивать эффективность алгоритма при различных параметрах, были получены следующие оценки для QH-AHP-алгоритма: $a_{21}=3,\ a_{31}=7,\ a_{32}=5,$ $S=53,\ \lambda_0=0{,}8,\ \lambda_1=0{,}01,\ \mu=30,\ \tau_{\max}=16.$ Для QH-B-алгоритма: $P(H_1)=0{,}83,\ P(H_2)=0{,}6,$

$P(H_3)=0,\ u_1=0{,}12,\ u_2=0{,}16,\ u_3=0{,}72,\ w_1=0{,}48,\ w_2=0{,}52,\ w_3=0{,}85,\ S=30,\ \lambda_0=0{,}4,\ \lambda_1=0{,}01,$ $\mu=27,\ \tau_{\max}=26.$

Полученные оптимальные значения параметров использовались в дальнейшем при исследовании скорости сходимости рассматриваемых алгоритмов оптимизации на тестовых функциях. Графики скорости сходимости для тестовых функций (средние значения целевой функции лидера за 10 запусков алгоритма) показаны на рис. 2.

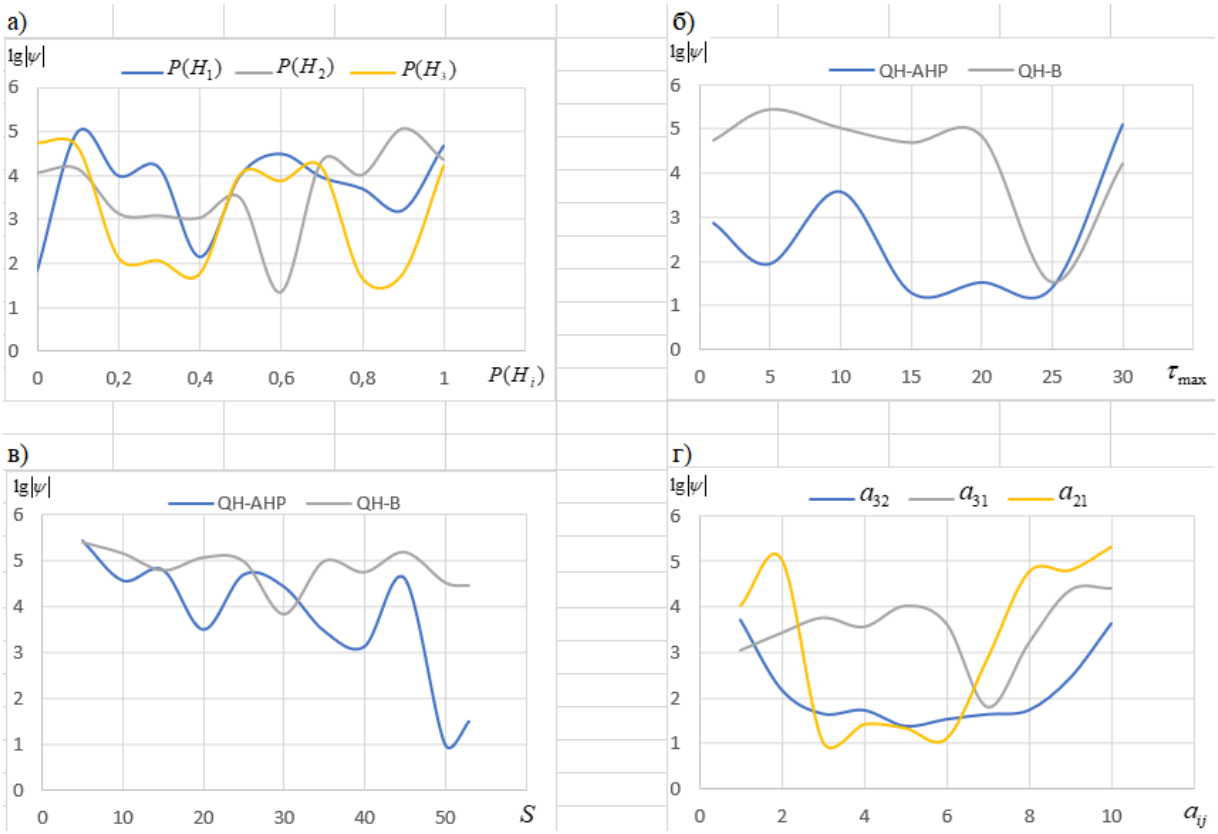


Рис. 1. Графики зависимости скорости сходимости алгоритмов от значений параметров: а) QH-B-алгоритма от $P(H_i)$; б) QH-AHP и QH-B-алгоритмов от $\tau_{\max}$; в) QH-AHP и QH-B-алгоритмов от S; г) QH-AHP-алгоритма от a_{ij}

В проведенном эксперименте во всех случаях квантовый алгоритм пресноводных гидр, основанный на методе анализа иерархий, показал лучшие результаты, чем алгоритм, использующий байесовский подход.

Заключение

Планирование событий, управление экономическими системами, проектирование производственных систем направлены на поиск наилучшего варианта для нахождения поставленной цели. Особенно сложны в решении задачи со сложным ландшафтом поверхности поиска, связанные с такими понятиями, как многоэкстремальность, овражность, многокритериальность [5]. Для эффективного решения сложных оптимизационных задач актуально использовать класс алгоритмов, называемый популяционными алгоритмами, в частности, можно рассматривать исследованный нами алгоритм пресноводных гидр.

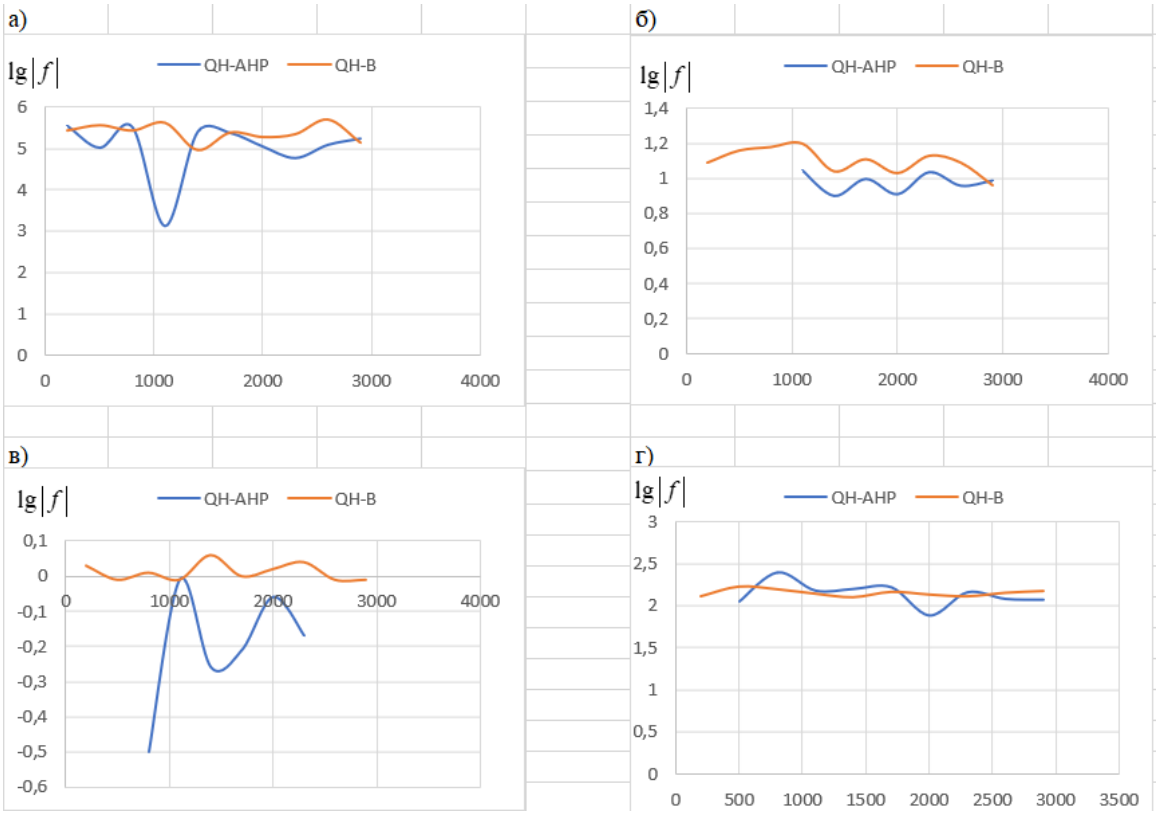


Рис. 2. Графики скорости сходимости на примере тестовых функций в зависимости от числа итераций:

а) f_1 *— Розенброка; б)* f_2 *— Дэвиса; в)* f_3 *— Гриванка; г)* f_4 *— Растригина*

Литература

1. Королев, А.С. Квантовая модификация алгоритма пресноводных гидр для решения задачи оптимизации / А. С. Королев, Д. В. Майков // Вестник ВГУ, серия: системный анализ и информационные технологии. – 2020. – №2. – С. 37–48.
2. Грошев, С.В. Мета-оптимизация популяционных алгоритмов многоцелевой оптимизации [Электронный ресурс] / С.В. Грошев, А.П. Карпенко // Интернет-журнал «Науковедение». – 2016. – Т. 8, №6. – С. 1–11. – Режим доступа: http://naukovedenie.ru/vol8-6.php. – (Дата обращения: 13.12.2016).
3. Разработка программного комплекса для синтеза популяционных алгоритмов. – Режим доступа: https://referat.yabotanik.ru. – (Дата обращения: 15.03.2015).
4. Генетические алгоритмы. – Режим доступа: http://www.gai.narod.ru. – (Дата обращения: 2003).
5. Тимофеева, О.П. Исследование популяционных алгоритмов в решении задач непрерывной оптимизации / О.П. Тимофеева, С.А. Неимущев, Л.И. Неимущева // Труды НГТУ им. Р.Е. Алексеева. – 2018. – №4. – С. 48–55.

УДК 004.4

РАЗРАБОТКА ИГРОВЫХ МЕХАНИК С ПРИМЕНЕНИЕМ WEB-ТЕХНОЛОГИЙ

С. А. Платонов

Воронежский государственный университет

Введение

В современном мире получили развитие технологии, которые позволяют заменить многие приложения, требующие установки на компьютер, на их аналоги в виде web-приложений.

Первая цель работы: выяснить насколько современные веб технологии подходят для создания карточных игр, в частности проекта “Свитки”. Проект представляет собой карточную игру типа дуэль, в которой два игрока разыгрывают карты для нанесения урона противнику. Другая цель – сделать выбор подходящих технологий и реализовать с их помощью основные игровые механики.

В связи с этим были решены следующие задачи:

1. Анализ фреймворка React JS.
2. Анализ хранилищ состояния.
3. Реализация отображения изменений в реальном времени.
4. Реализация основных игровых механик (нанесение урона и разыгрывание карт).

1. Анализ React JS

React JS — это UI-библиотека, созданная Facebook, для упрощения и оптимизации работы с элементами DOM (document object model).

1.1 Преимущества

1. Фреймворк продвигает компонентный подход, смысл которого в разделении всего сайта на отдельные независимые части с целью их дальнейшего переиспользования, что позволяет разбить весь интерфейс, а также всю логику для него, на модули, хранящиеся в отдельных файлах.

2. Высокая популярность и растущее сообщество разработчиков дают широкий выбор дополнительных инструментов, которые могут понадобиться при разработке нетривиальной игровой логики, а также увеличивает вероятность того, что проблемы, с которыми можно столкнуться при разработке, уже решены.

3. Наличие виртуальной DOM, которая оптимизирует работу по изменению и сравнению с браузерным DOM-деревом, что гарантирует стабильную работу при высоких нагрузках.

4. Маленький вес самой библиотеки, что в совокупности с предыдущим пунктом, дает уверенность, в том, что пользователь не столкнется с проблемами при использовании.

1.2 Недостатки

1. Документация, которая не покрывает работу более сложных моментов библиотеки, с которыми потенциально можно столкнуться при реализации игровых механик.

2. Отсутствие встроенных решений для базовых моментов, таких как роутинг и хранение состояния, что усложняет выбор инструментов.

2. Анализ хранилищ состояния

При разработке приложения с потенциально нетривиальной логикой возникает необходимость в применении хранилища состояний, которое представляет собой "глобальную переменную", в которую можно положить и из которой можно взять из любой точки приложения. Без хранилища состояний из-за хранения HTML элементов в виде дерева придется передавать информацию от родителя к ребенку через несколько уровней, что замедлит работку приложения и заметно усложнит логику. Два самых популярных хранилища состояний — это Redux и MobX

2.1 Redux

Библиотека создана в 2015 году русским разработчиком Даниилом Абрамовым и Эндрю Кларком. Является первым хранилищем состояний и самым популярным среди конкурентов. Но заметно усложняет логику и требует написания большого количества кода для работы.

2.2 MobX

MobX разработан с оглядкой на Redux. Более легковесное и требующее минимального количества кода хранилище состояний. В коде выглядит как обычный объект, что упрощает логику работы с ним.

После анализа этих двух библиотек был выбран MobX

3. Реализация отображения изменений в реальном времени

Ключевой функцией проекта “Свитки” является игра онлайн, что подразумевает отображение действий одного игрока у другого. В такой ситуации обычных HTTP запросов недостаточно, т. к. они срабатывают один раз и отправляются с фронтенд части. Для решения подобных проблем используется протокол WebSocket, который устанавливает долгосрочное соединение с сервером. Для упрощения и ускорения работы был выбран облачный сервис Firebase, который предоставляет обертку над WebSocket, с более удобным API, что дает возможность избежать написание поддержки WebSocket для приложения с нуля.

4. Реализация основных игровых механик

Для реализации проекта “Свитки” необходимо разработать две основные механики: нанесение урона и розыгрыш карт.

4.1 Механика нанесения урона

Для реализации данной механики было необходимо спроектировать структуру данных для игрока (рис. 1), которая содержит выложенные на стол карты, карты в руке и id выбранной для розыгрыша карты,

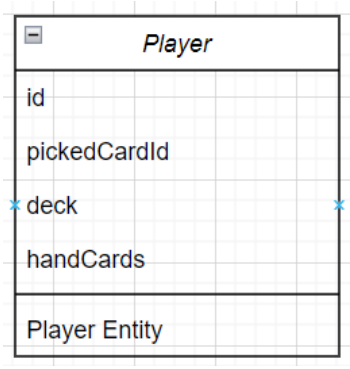


Рис.1

структуру персонажа игрока (рис. 2)

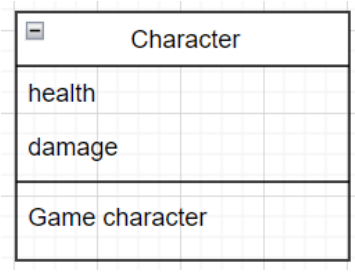


Рис.2

и структуру карты (рис. 3)

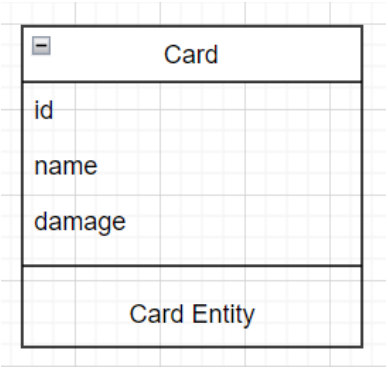


Рис.3

Для переноса карт из руки на стол была использована функция drag-and-drop, после нажатия на карту обновляется информация в структуре персонажа, и при нажатии на вражеского персонажа ему наносится урон

Заключение

Результатом работы стало выявление положительных и отрицательных сторон современных web-технологий, понимание того, насколько они подходят для замены стандартных приложений, а также реализация основных игровых механик с помощью наиболее удобных для данного приложения.

Литература

1. Документация React. – Режим доступа: https://legacy.reactjs.org (Дата обращения: 11.04.2023).

2. Плюсы и минусы React – Режим доступа: https://www.knowledgehut.com/blog/web-development/pros-and-cons-of-react (Дата обращения: 11.04.2023).

3. Redux vs MobX – Режим доступа: https://blog.logrocket.com/redux-vs-mobx (Дата обращения: 11.04.2023).

УДК (519.61)

ИНТЕРПОЛЯЦИОННЫЙ СМЫСЛ МЕТОДА МАСШТАБИРОВАНИЯ И КВАДРИРОВАНИЯ

И. С. Плотников

Воронежский государственный университет

Введение

Для нахождения решений систем обыкновенных дифференциальных уравнений [7] во многих случаях возникает необходимость построения матричной экспоненты. Во-первых, если исходная система является линейной и имеет постоянные коэффициенты, то решение непосредственно выражается через матричную экспоненту. Во-вторых, если в системе имеется явно выраженная линейная часть, то обращение этой линейной части (сводящееся к построению матричной экспоненты) может существенно упростить решение всей нелинейной системы.

Для матриц размера больше 3x3 матричную экспоненту обычно вычисляют приближенно [3, 8] компьютерными средствами. Известно большое количество алгоритмов ее нахождения. В настоящее время считается, что наиболее эффективным методом вычисления матричной экспоненты является метод масштабирования и квадрирования.

Настоящая статья посвящена исследованию метода масштабирования и квадрирования. А именно, описывается его интерпретация с интерполяционной точки зрения. Понимание интерполяционного смысла данного метода позволяет, например, получить оценки его точности [10].

1. Определение матричной экспоненты

Метод масштабирования и квадрирования применим только к линейным системам с постоянными коэффициентами, то есть к системам, имеющим вид

$$x'(t) = \mathbf{A}x(t),$$

где $\mathbf{A}$ – квадратная матрица.

Матричной экспонентой этой матрицы называют сумму ряда Тейлора

$$e^{\mathbf{A}t} = I + \mathbf{A}t + \frac{\mathbf{A}^2t^2}{2!} + \ldots = \sum_{k=0}^{\infty} \frac{\mathbf{A}^k t^k}{k!}.$$

Таким образом, задача состоит в вычислении суммы бесконечного ряда, который можно приблизительно найти методами численного анализа.

2. Аппроксимация Паде

Помимо приближения рядом Тейлора, в математике используют приближение, называемое аппроксимацией Паде [6]. Напомним определение аппроксимации Паде.

Пусть функция f в окрестности нуля раскладывается в степенной ряд Тейлора

$$f(z)=\sum_{i=0}^{\infty}c_i z^i.$$

Аппроксимацией Паде называют рациональную функцию вида

$$\prod_{[L/M]}(z)=\frac{p_{LM}(x)}{q_{LM}(x)}=\frac{a_0+a_1 z+\ldots+a_L z^L}{b_0+b_1 z+\ldots+b_M z^M},$$

обладающую свойством

$$f(z)=\sum_{i=0}^{\infty}c_i z^i=\frac{a_0+a_1 z+\ldots+a_L z^L}{b_0+b_1 z+\ldots+b_M z^M}+O(z^{L+M+1}).$$

Найти аппроксимацию Паде – значит подобрать коэффициенты $a_0,a_1,\ldots,a_L,b_0,b_1,\ldots,b_M$, для которых это равенство выполняется. Заметим, что здесь число независимых параметров $a_0,a_1,\ldots,a_L,b_0,b_1,\ldots,b_M$ равно $L+M+1$ (полагаем, что $b_0=1$ во избежание неопределенности). Известны различные методы, позволяющие найти $a_0,a_1,\ldots,a_L,b_0,b_1,\ldots,b_M$. Для случая $f(z)=e^z$ формулы для коэффициентов $a_0,a_1,\ldots,a_L,b_0,b_1,\ldots,b_M$ можно найти в [6].

Описываемый в следующем параграфе метод масштабирования и квадрирования использует аппроксимацию Паде для вычисления матричной экспоненты в случае маленькой по норме матрицы $\mathbf{A}$. Отметим сразу преимущества в использовании в этом случае аппроксимации Паде вместо тейлоровского разложения:

1. Улучшенная точность: аппроксимация Паде дает лучшую точность приближения, чем многочлен Тейлора с тем же числом коэффициентов.
2. Лучшая стабильность: аппроксимация Паде менее чувствительна к ошибкам округления, потому что в то время как многочлен Тейлора состоит из суммы степенных функций, аппроксимация Паде представляет собой дробь, числитель и знаменатель которой – многочлены. В результате численные результаты, полученные с использованием аппроксимации Паде, обычно более точны, чем те, которые получены с использованием многочлена Тейлора.
3. Быстрая сходимость: аппроксимации Паде обычно сходятся быстрее ряда Тейлора, потому что они обеспечивают лучшее приближение функции с меньшим числом членов. Дробные функции позволяют лучше учитывать особенности функции, такие как асимптотическое поведение и колебания.
4. Лучшая аппроксимация некоторых конкретных функций: в ряде случаев аппроксимация Паде дает лучшее приближение, особенно, если речь идет о колеблющихся функциях, например, она лучше приближает экспоненциальную функцию около мнимой оси.

Из недостатков стоит отметить необходимость оптимального подбора параметров L и M аппроксимации Паде – правильный их выбор может повлиять на точность метода

масштабирования и квадрирования. Большие значения L и M положительно скажутся на теоретической точности, но могут заметно увеличить время вычислений и ошибки округления.

3. Метод масштабирования и квадрирования

Метод масштабирования и квадрирования основан на использовании соотношения

$$e^{\mathbf{A}} = \left(e^{\frac{\mathbf{A}}{\sigma}} \right)^{\sigma}$$

для $\mathbf{A} \in C^{n\times n}$ и $\sigma \in \mathrm{N}$, а также на том факте, что $e^{\mathbf{A}}$ при малых по норме матрицах $\mathbf{A}$ может быть приближена аппроксимацией Паде с достаточно высокой точностью. Возьмем в качестве σ степень числа 2 ($\sigma = 2^s$) и приблизим:

$$e^{\frac{\mathbf{A}}{2^s}} \approx r\left(\frac{\mathbf{A}}{2^s}\right),$$

где r – аппроксимация Паде, а затем скажем, что

$$e^{\mathbf{A}} \approx \left[r\left(\frac{\mathbf{A}}{2^s}\right) \right]^{2^s}.$$

Вычисления по данной формуле можно производить достаточно быстро, поскольку нахождение степени 2^s сводится к последовательному возведению в квадрат s раз.

Существенно облегчает вычисления тот факт, что известны явные формулы для числителя и знаменателя аппроксимации Паде:

$$p_{LM}(z) = \sum_{j=0}^{L} \frac{(L+M-j)!L!}{(L+M)!(L-j)!} \frac{z^j}{j!},$$

$$q_{LM}(z) = \sum_{j=0}^{M} \frac{(L+M-j)!L!}{(L+M)!(M-j)!} \frac{(-z)^j}{j!}.$$

4. Интерполяционный смысл метода масштабирования и квадрирования

В вычислительной математике наиболее эффективным методом приближенного вычисления функций считается интерполяция. Задачу вычисления матричной экспоненты, безусловно, можно интерпретировать как задачу приближенного вычисления некоторой функции от матрицы, а именно, $\left[r\left(\frac{z}{2^s}\right) \right]^{2^s}$. Тем не менее, в своей первоначальной постановке метод масштабирования и квадрирования не является интерполяционным. Целью настоящей работы является выяснение интерполяционного смысла метода масштабирования и квадрирования.

Рассмотрим приближенную формулу

$$e^z \approx \left[r\left(\frac{z}{2^s}\right) \right]^{2^s}.$$

Если удастся найти точки $z_1, z_2, ..., z_n$ в которых приближенное равенство превращается в точное

$$e^z = \left[r\left(\frac{z}{2^s}\right) \right]^{2^s},$$

то функцию $\left[r\left(\frac{z}{2^s}\right) \right]^{2^s}$ можно интерпретировать как рациональную функцию, интерполирующую функцию e^z в точках $z_1, z_2, ..., z_n$. Таким образом, задача нахождения интерполяционного смысла сводится к нахождению корней уравнения

$$e^z = \left[r\left(\frac{z}{2^s}\right) \right]^{2^s}.$$

Трудность в решении этого уравнения заключается в том, что оно трансцендетное (а не полиномиальное) и поэтому может быть решено только численно. Более того, из-за высокой степени многочленов, стоящих в числителе и знаменателе, для получения решения с приемлемой точностью приходится решать это уравнение с повышенным числом значащих цифр.

Теория рекомендует [3] в качестве параметров брать $L = M = 13$. Величину параметра s рекомендуется подбирать так, чтобы $\left\| \mathbf{A}^{2^s} \right\| \le \frac{1}{2}$. Таким образом, реальное уравнение, возникающее в методе масштабирования и квадрирования принимает вид

$$e^z = \left[\frac{\sum_{j=0}^{L} \frac{(L+M-j)!L!}{(L+M)!(M-j)!} \frac{(-z)^j}{j!}}{\sum_{j=0}^{M} \frac{(L+M-j)!L!}{(L+M)!(M-j)!} \frac{(-z)^j}{j!}} \right]^{2^s}.$$

Мы ограничимся рассмотрением случая $s = 3$, поскольку это наиболее сложный случай, доступный для реальных вычислений.

Для решения этого уравнения в работе использовался метод Ньютона. Как известно, для применения метода Ньютона требуется начальное приближение. Кроме того, мы ожидаем, что корней будет несколько, поэтому нужно брать много различных начальных приближений (по крайней мере, по одному на каждый корень). В связи с этим возникают вопросы:

1. Как выбирать начальное приближение?
2. Как убедиться в том, что найдены все корни?

5. Выбор начального приближения

Итак, рассмотрим случай $s = 3$, $\sigma = 8$, $L = M = 13$. Имеем следующее уравнение:

$$e^z = \left\lceil r_{13/13}\left(\frac{z}{2^s}\right)\right\rceil^{2^s}.$$

Поскольку мы заранее не знаем, где находятся корни, то сперва зададим начальные приближения случайным образом на комплексной плоскости. Результат вычислений показан на рис. 1. Начальные точки изображены зеленым, а полученные численно корни уравнения – красным. Можно заметить, что корни группируются около нулей числителя (в дальнейшем мы увидим, что пока они найдены не все).

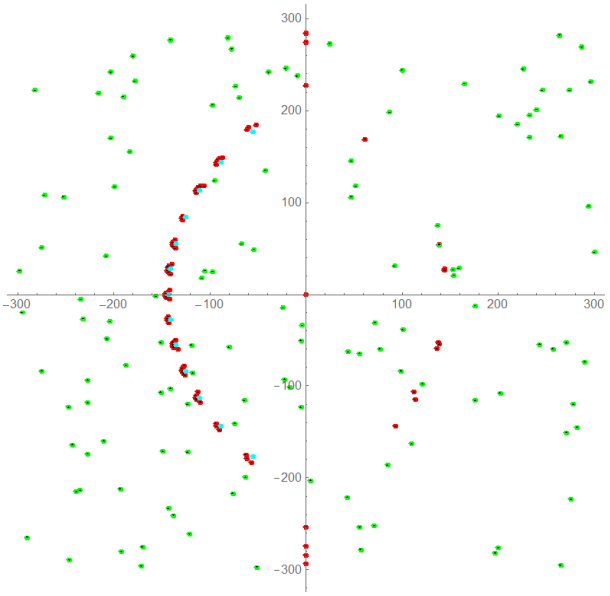


Рис. 1. Начальные условия и численные решения

Чтобы ускорить нахождение корней, улучшить точность и добиться того, чтобы корни не терялись, вместо случайных начальных точек разумно брать точки, находящиеся около нулей числителя. Чтобы ни один из корней трансцендетного уравнения не пропустить, в качестве начальных точек бралось большое количество точек на окружностях небольшого радиуса с центрами в нулях числителя. В качестве радиуса бралось значение 10.5 (оно было подобрано экспериментально). Чтобы контролировать процесс вычислений, также стрелками показано, из какой начальной точки в какой корень метод Ньютона нас приводит. Результат показан на рис. 2.

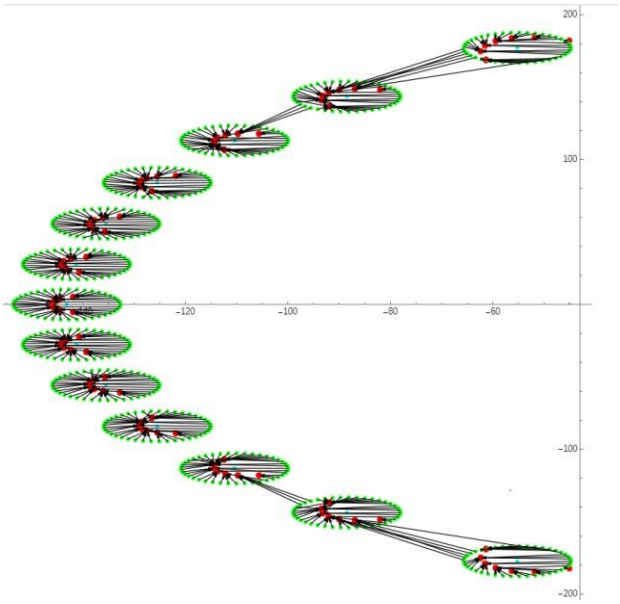


Рис. 2. Начальные условия и численные решения

Из рисунка 2 видно, что около каждого нуля числителя возникает семь численно полученных точек интерполяции. Очевидно, число семь есть $2^s - 1$. Таким образом, найдено $L \times (2^s - 1) = 13 \times 7 = 91$ интерполяционных условий, которым удовлетворяет используемая рациональная функция. Знание интерполяционных условий позволяет, например, выписывать оценки точности приближений [8, 10].

Вычисления проводились в пакете Wolfram Mathematica [9].

Заключение

Таким образом, в данной работе выявлен интерполяционный смысл метода масштабирования и квадрирования и численно найдены точки интерполяции, неявно присутствующие в методе масштабирования и квадрирования. Исследование метода масштабирования и квадрирования может использоваться для улучшения процесса вычисления матричных экспонент, например, оценки точности полученного результата.

Работа выполнена под руководством В.Г. Курбатова, профессора кафедры САиУ Воронежского госуниверситета.

Литература

1. Fasi, M. An arbitrary precision scaling and squaring algorithm for the matrix exponential / M. Fasi, N. J. Higham // SIAM J. Matrix Anal. Appl. -2019. – Vol. 40, no. 4 – P. 1233-1256.

2. Higham, N. J. Functions of matrices: theory and computation / N. J. Higham. – Philadelphia,PA : Society for Industrial and Applied Mathematics (SIAM), 2008. – xx+425 p. – ISBN: 978-0-89871-646-7.

3. Higham, N. J. The scaling and squaring method for the matrix exponential revisited / N. J. Higham // SIAM Rev. – 2009. – Vol. 51, no. 4. – P. 747-764.

4. Koikari, S. An error analysis of the modified scaling and squaring method / S. Koikari // Comput. Math. Appl. – 2007. – Vol. 53, no. 8. – P. 1293 1305.

5. Skaflestad, B. The scaling and modified squaring method for matrix functions related to the exponential / B. Skaflestad, W. M. Wright // Appl. Numer. Math. – 2009. – Vol. 59, no. 3-4. P. 783 799.

6. Бейкер мл., Д. Аппроксимация Паде / Дж. Бейкер мл., П. Грейвс-Моррис. – М. : Мир, 1986. – 502 с.

7. Боровских, А. В. Лекции по обыкновенным дифференциальным уравнениям / А. В. Боровских, А. И. Перов. – М. – Ижевск: Регулярная и хаотическая динамика, 2004. – 540 с. – ISBN: 5-93972-327-6.

8. Курбатов, В. Г. Вычислительные методы спектральной теории : учебное пособие / В. Г. Курбатов, И. В. Курбатова. - Воронеж : Издательский дом ВГУ, 2019. – 323 с. – ISBN: 978-5-9273-2824-6.

9. Курбатов, В. Г. Пакет «Математика» в прикладных научных исследованиях : учебное пособие / В. Г. Курбатов, В. Е. Чернов. – Воронеж : Издательский дом ВГУ, 2016. – 240 с. – ISBN: 978-5-9273-2297-8.

10. Kurbatov, V. G. An estimate of approximation of a matrix-valued function by an interpola- tion polynomial / V. G. Kurbatov, I. V. Kurbatova // Eurasian Math. J.— 2020.— Vol. 11, no. 1.— P. 86–94.

УДК 004.023

АНАЛИЗ И СРАВНЕНИЕ МЕТОДОВ РЕШЕНИЯ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА

Н. В. Покатаев

Воронежский государственный университет

Введение

Задача маршрутизации транспортных средств (Vehicle routing problem, VRP) – одна из актуальных задач современного мира. Несмотря на свою актуальность, она широко изучалась с момента ее формулировки в конце 1950-х годов. Ее решение используется для определения наиболее эффективных маршрутов движения парка транспортных средств с целью обслуживания определенного круга клиентов или местоположений.

Цель задачи состоит в том, чтобы минимизировать затраты на общую эксплуатацию при одновременном удовлетворении требований заказчика, например временных рамок или требования к количеству точек назначения. В качестве затрат могут рассматриваться затраты на топливо, рабочую силу и количество используемых транспортных средств.

За время изучения было разработано множество методов решения. К ним относятся точные методы, эвристика и метаэвристика. Точные методы обычно используются для задач небольшого размера, в то время как эвристика и метаэвристика используются для более крупных. Например, эвристика предназначена для быстрого генерирования выполнимых решений, в то время как метаэвристика для поиска решений, близких к оптимальным.

Задача является сложной из–за ее комбинаторной природы и наличия множества целей. Сложность сильно возрастает с увеличением количества транспортных средств, клиентов и накладываемых ограничений. Таким образом, важно, чтобы данные и цели были точно определены, а задача сформулирована должным образом.

В этой статье будет рассмотрена разновидность задачи маршрутизации транспортных средств с ограничением по грузоподъемности (Capacitated VRP, CVRP) и различные методы решения, которые были разработаны за эти годы.

1. Постановка задачи

Задача CVRP может быть представлена в виде модели линейного целочисленного программирования, которая является симметричной. Важно отметить, что это достаточно популярное представление, однако в некоторых работах рассматривается ассиметричное [1].

Пусть G полный направленный граф:

$$G = (V, H, c), \tag{1}$$

V множество вершин:

$$\mathrm{V} = \{0, 1, 2, \ldots, n\}, \tag{2}$$

H множество дуг, идущих из некоторой вершины i в j, но не входящих в саму себя:

$$H = \{(i, j) : i, j \in V, i \neq j\}, \tag{3}$$

Узел 0 представляет депо для автопарка из P автомобилей вместимости Q. Остальные n узлов представляют географически рассредоточенных клиентов.

Каждый клиент имеет определенный положительный спрос:

$$d_i \le Q, i \in V - \{0\}, \tag{4}$$

Неотрицательная стоимость поездки c_{ij} ассоциирована с каждой дугой $(i,j) \in H$. Матрица стоимости является симметричной, т. е. $c_{ij} = c_{ji}$для всех $i,j \in V, i \neq j$ и удовлетворяет неравенству треугольников $c_{ij} + c_{jk} \geq c_{ik}$ для все $i,j,k \in V$.

Тогда минимальное число автомобилей для обслуживания всех клиентов может быть представлено в виде:

$$\frac{\sum_{i=1}^{n} d_i}{Q}, \tag{5}$$

Тогда модель линейного целочисленного программирования задачи CVRP примет следующий вид [2]:

$$Min \sum_{r=1}^{p} \sum_{i=0}^{n} \sum_{j=0, i \neq j}^{n} c_{ij}\, x_{rij}, \tag{6}$$

$$\sum_{r=1}^{p} \sum_{i=0, i \neq j}^{n} x_{rij} = 1, \quad \forall j \in \{1,\ldots,n\}, \tag{7}$$

$$\sum_{j=1}^{n} x_{r0j} = 1, \quad \forall r \in \{1,\ldots,p\}, \tag{8}$$

$$\sum_{i=0, i \neq j}^{n} x_{rij} = \sum_{i=0}^{n} x_{rji}, \quad \begin{matrix} \forall j \in \{0,\ldots,n\}, \\ \forall r \in \{1,\ldots,p\}, \end{matrix} \tag{9}$$

$$\sum_{i=0}^{n} \sum_{j=1, i \neq j}^{n} d_j x_{rij} \le Q, \quad \forall r \in \{1,\ldots,p\}, \tag{10}$$

$$\sum_{r=1}^{p} \sum_{i \in S} \sum_{j \in S, i \neq j} x_{rij} \le |S| - 1, \quad \forall S \in \{1,\ldots,n\}, \tag{11}$$

$$x_{rij} \in \{0,1\}, \quad \begin{matrix} \forall r \in \{1,\ldots,p\}, \\ i,j \in \{0,\ldots,n\}, \\ i \neq j. \end{matrix} \tag{12}$$

Целевая функция (6) минимизирует общую стоимость поездки. Модельные ограничения (7) являются ограничениями степени и гарантируют, что каждого клиента посещает ровно одно транспортное средство. Ограничения потока (8) и (9) гарантируют, что каждое транспортное средство может покинуть склад только один раз, а количество транспортных средств, прибывающих к каждому клиенту и въезжающих на склад, равно количеству выезжающих транспортных средств. В ограничениях (10) указаны ограничения по вместимости, обеспечивающие, чтобы сумма запросов клиентов, посещенных по маршруту, была меньше или равна вместимости транспортного средства выполнение услуги. Ограничения на исключение подтурниров (11) гарантируют, что решение не содержит циклов, отключенных от хранилища. Остальные обязательные ограничения (12) определяют области определения переменных. Эта модель известна как трехиндексная формула расхода транспортного средства. Число неравенств ограничений на устранение подтурниров растет экспоненциально с увеличением числа узлов.

2. Методы решения задачи

2.1 Параллельный алгоритм накопления

Данный алгоритм был представлен Кларком и Райтом в 1964 году. Его идея основана на сокращении расстояния, пройденного транспортным средством, посредством некоторой функции сбережения [15].

Введем обозначения. $S(P,Q)$ Функция сбережения Кларка и Райта (The Clarke and Wright savings function) двух клиентов P и Q находящихся на расстоянии d_{OP} и d_{OQ} относительно депо 0*:*

$$S(P,Q) = d_{OP} + d_{OQ} - d_{PQ}, \tag{13}$$

Пусть клиент C – новая часть пути, между клиентами A и B. Тогда функция общего сбережения $SAV_c(A,B)$:

$$ST_C(A,B) = d_{AC} + d_{BC} - d_{AB}, \tag{14}$$

$$SAV_c(A,B) = 2d_{OC} - ST_C(A,B), \tag{15}$$

Тогда лучшее расположение клиента C может быть вычислено, как:

$$ST_C(I,J) = Min\{ST_C(A,B)\}, \tag{16}$$

$$SAV_K(L,M) = Max\{SAV_C(I,J)\}, \tag{17}$$

Таким образом алгоритм построения маршрута можно представить в виде 3 повторяющихся шагов:

1. Вычисляется функция сбережения (16) для всех не посещённых узлов;
2. Выполняется поиск пары с наибольшим значением функции сбережения (17);
3. Решение добавляется к маршруту.

Алгоритм заканчивает свою работу в тот момент, когда не остается вершин, доступных для посещения.

2.2. Алгоритм развертки

Данный алгоритм был разработан в 1974 году. Он имеет эвристическую направленность и используется для определения наилучшего маршрута для набора транспортных средств, следующих из одного места в другое [16].

Введем обозначение. Пусть N – число локаций, включая депо (обозначим его номером 1). $Q(I)$– требования в локации I. $(X(I),Y(I))$ – прямоугольный координаты I локации. C вместимость каждого транспортного средства. D – максимальная расстояние, на которое может выехать машина из депо. $R(I)$ – радиус из депо в локацию I. $A(I,J)$ – расстояние между локациями I и J. $An(I)$ – угол в полярных координатах для I

локации, определенный как: $\mathrm{An(I)} = \tan^{-1}\frac{Y(I)-Y(1)}{X(I)-X(1)}$, где

$$\begin{cases} -\pi < An(I) < 0, \text{если } Y(I) - Y(1) < 0, \\ 0 \le An(I) \le \pi, \text{если } Y(I) - Y(1) \ge 0. \end{cases}$$

Также введем следующие ограничения:

$$Q(I) \le C, \qquad \forall I \in \{1, \ldots, N\}, \tag{18}$$

$$A(I,J) > 0, \qquad \begin{matrix} \forall I \in \{1, \dots, N\}, \\ \forall J \in \{1, \dots, N\}, \\ I \neq J. \end{matrix} \tag{19}$$

$$A(I,I) = 0, \tag{20}$$

$$A(I,1) + A(1,I) \leq D \qquad \forall I \in \{1, \dots, N\}. \tag{21}$$

Предположим, что локации пронумерованы в соответствии с размером их угла в полярных координатах. Тогда $An(I) < An(I+1), \forall I \in \{1, \dots, N\}$. Если существует I и J такие, что $An(I) = An(J)$, тогда $I < J$, если $A(1,I) < A(1,J)$. Благодаря этому условию, сохраняется уникальность порядка.

Ход работы алгоритма состоит из двух частей: прямая и обратная развертка. Алгоритм прямой развертки разбивает маршруты, начиная с локации с наименьшим углом (локация 2). Первый маршрут строится из локаций $2,3, \dots, J$, где J – последняя точка, которую можно добавить, не превышая вместимости транспортного средства или ограничения по пройденному расстоянию. Аналогично формируются остальные маршруты. Когда все локации будут рассмотрены, общее расстояние будет равно сумме расстояний по каждому построенному маршруту.

После этого совершаются попытки заменить одну локацию K на одну или несколько локаций в маршруте $K+1$ для $K = 1,2, \dots, m-1$, где m – число сформированных маршрутов. Локация, которую необходимо удалить из маршрута K, получается путем минимизации функции радиуса $R(I)$ и угла $An(I)$ каждой локации на маршруте K. Первая локация P, рассматриваемая для включения в маршрут K, — это локация в маршруте $K+1$, ближайшая к последней локации, добавленной в маршрут. Вторая, рассматриваемая для включения в маршрут K, — это локация на маршруте $K+1$, ближайшем к точке P. Если по этой схеме к маршруту $K+1$ добавляется одна или несколько локаций, то следующие локации на маршруте $K+1$ также проверяются, чтобы убедиться, что они могут быть включены в K. Этот процесс продолжается до тех пор, пока J не станет последней локацией в маршруте K.

Вторая часть или алгоритм обратной развертки, похож на прямую, за исключением того, что он формирует маршруты в обратном порядке. Изначально локация l содержит $N, N-1, N-2, L$, маршрут 2 содержит $L-1, L-2, \dots, M$ и так далее. В большинстве случаев эти две процедуры дают разные маршруты и, следовательно, разные минимальные общие расстояния. Наименьший из этих двух минимумов является наилучшим приближенным решением.

2.3 Последовательный алгоритм построения маршрута

Разработка данного алгоритма берет начало в 1976 году с разработки «Обобщенного критерия экономии» [17]. Это эвристический алгоритм, который изначально был разработан для решения задачи Коммивояжера, но может быть применен для решения поставленной задачи.

$\mathrm{MS}(P,Q)$ Критерий Гаскелла [6] представляет собой модифицированную функцию сбережения:

$$MS(P,Q) = S(P,Q) - \pi d_{PQ}, \tag{22}$$

Благодаря π достигается построение более радиальных маршрутов. Важно отметить, что в отличии от (1) выражение (6) необязательно неотрицательно. Модифицированная функция

общего сбережения примет вид:

$$MST_C(A, B) = d_{AC} + d_{BC} - \mu d_{AB}, \quad (23)$$

$$MSAV_c(A, B) = \lambda d_{OC} - MST_C(A, B), \quad (24)$$

А наилучшее расположение клиента:

$$MST_C(I, J) = Min\{MST_C(A, B)\}, \quad (25)$$

$$MSAV_K(L, M) = Max\{MSAV_C(I, J)\}, \quad (26)$$

Значения параметров λ и μ представлены в таблице (табл. 1)

Табл. 1 Таблица зависимости критериев от значений λ и μ.

λ	μ	Критерий
2	1	Обобщенный критерий Кларка и Райта
0	1	Критерий минимальная «нагрузка»
[1; 2]	$\lambda - 1$	Критерий Гаскелла
$+\infty$	$\mathbb{N}^*$	Критерий в порядке удаления от депо
0	0	Критерий близости к двум ближайшим соседям

Ход выполнения аналогичен алгоритму параллельного накопления.

3.Тестирование и сравнение результатов

Тестирование и сравнение алгоритмов будет проводится на заранее подготовленных тестовых примерах. VeRyPy [7] – является пакетом эвристических алгоритмов с открытым исходным кодом, содержащим все представленные в тексте настоящей работы алгоритмы. В качестве входных данных будут использованы файлы спецификации TSPLIB95.

В ходе тестирования были получены следующие результаты (рис. 1).

Рис 1. Результаты тестирования. Время выражено в секундах, стоимость в условных единицах

В тестировании участвовало 3 задачи, различающиеся по числу локаций: 51, 72 и 135, соответственно. Из графиков видно, что последовательный алгоритм построения маршрутов показал лучший результат во всех тестах, кроме первого. Однако этот алгоритм требовал больше всего временных затрат. Алгоритм развертки справился с задачей хуже всего, относительно качества полученного решения и затраченного на его получение времени. Это

обусловлено тем фактом, что входные данные были спроектированы для алгоритмов другого класса. Алгоритм параллельного накопления оказался наилучшим из рассмотренных.

Заключение

В статье представлен краткий обзор различных методов, доступных для решения задачи маршрутизации транспорта. Каждый из этих методов имеет большую историю, успел зарекомендовать себя в научном сообществе, а также имеет свои преимущества и недостатки, исходя из которых необходимо выбирать, какой из них является наиболее подходящим для конкретных входных данных.

Список литературы

1. Beresneva E.N., Avdoshin S. Analysis of Mathematical Formulations of Capacitated Vehicle Routing Problem and Methods for their Solution // Proceedings of the Institute for System Programming of RAS. 2018.
2. Borcinova Z. Two models of the capacitated vehicle routing problem // Croatian Operational Research Review. 2017.
3. Clarke G., Wright J.W. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points // Operations Research. 1964. №12.
4. Gillett B.E., Miller L.R. A Heuristic Algorithm for the Vehicle-Dispatch Problem // Operations Research. 1974. №22.
5. Mole R.H., Jameson S.R. A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion // Operational Research Quarterly. 1976. №27.
6. Gaskell T.J. Bases for Vehicle Fleet Scheduling // OR. 1967. №18.
7. Github – VeRyPy URL: https://github.com/yorak/VeRyPy (дата обращения: 07.04.2023).

УДК 004

ПРАВОВОЙ КОНСУЛЬТАНТ ДЛЯ ГРУЗОПЕРЕВОЗЧИКОВ

О.В. Поливцева

Воронежский государственный университет

Введение

Автомобильный транспорт выполняет функции по обеспечению производства, обращению продукции промышленности и сельского хозяйства, обеспечению нужд капительного строительства, удовлетворению потребностей граждан в перевозках. Основная деятельность автомобильного транспорта заключается в перевозках грузов, пассажиров, багажа и почты. Регулирование транспортных отношений осуществляется Конституцией РФ [1], Гражданским Кодексом РФ [2], транспортным уставом [3].

Перевозка — это перемещение на основании договора грузов, пассажиров, багажа в пространстве с помощью транспортных средств [4]. Сама перевозка включает комплекс действий и операций, выполняемых в определенной последовательности и объединенных в соответствующие этапы. Можно выделить следующие этапы перевозки:

1) подача заявки (заказа) на перевозку груза и принятие по ней решения;
2) подготовка груза и транспортного средства к перевозке;
3) прием груза, его погрузка;
4) оформление необходимых транспортных документов, подписание транспортной накладной;
5) перевозка груза;
6) выдача груза, его вывоз.

Среди всех средств перевозок автомобильный транспорт является самым массовым и публичным. Его качественное и эффективное правовое регулирование имеет важное значение страны. Но, к сожалению, не все сотрудники, связывающие свою деятельность с грузоперевозками, владеют достаточными знаниями в области законодательства. Это может привести не только к нарушениям законов и ответственности, но и к нежелательному сотрудничеству с мошенниками.

В общем доступе сейчас много справочных правовых систем, таких как «Консультант Плюс», «Гарант», «Кодекс», «Референт» и ряд других. Они предназначены для качественного и оперативного снабжения правовой информацией юристов, бухгалтеров, руководителей. Но для обычных пользователей, которые мало взаимодействуют с программным обеспечением, такие системы могут показаться очень сложными и неудобными.

Актуальность работы заключается в обеспечении доступности законодательства для водителей и экспедиторов. Для этого необходим простой в использовании правовой консультант для грузоперевозчиков.

1. Функциональность разработки

Платформа «Правовой консультант для грузоперевозчиков» реализована в виде web-сайта.

Возможности узкоспециализированы и направлены на деятельность в сфере грузовых перевозок автомобильным транспортом, с целью облегчить поиск нужной правовой информации водителям.

Пользователи сайта делятся на две категории: зарегистрированные и

незарегистрированные.

Лица, не имеющие регистрацию, могут получить информацию о возможном перегрузе и о том, кто какую ответственность за него несут.

Также можно посмотреть актуальные изменения правил дорожного движения. На сайте собраны изменения законодательства относящиеся к правилам движения грузовых автомобилей и непосредственно к осуществлению грузоперевозок.

Пользователям предоставляется возможность ознакомиться с документами, относящимся к правилам грузоперевозок, таким как: заключение договора перевозки груза, предоставление транспортных средств и контейнеров, предъявление и прием груза для перевозки, определение массы груза, сроки доставки, выдача груза и так далее.

В разделе «Типовые вопросы» можно получить информацию из судебной практики по актуальным вопросам грузоперевозок.

Зарегистрированные пользователи имеют больше возможностей, чем незарегистрированные (рис. 1).

Кроме тех разделов, которые доступны лицам без регистрации, у них появляются права на обращение к специалистам за юридической консультацией. А также они могут отследить

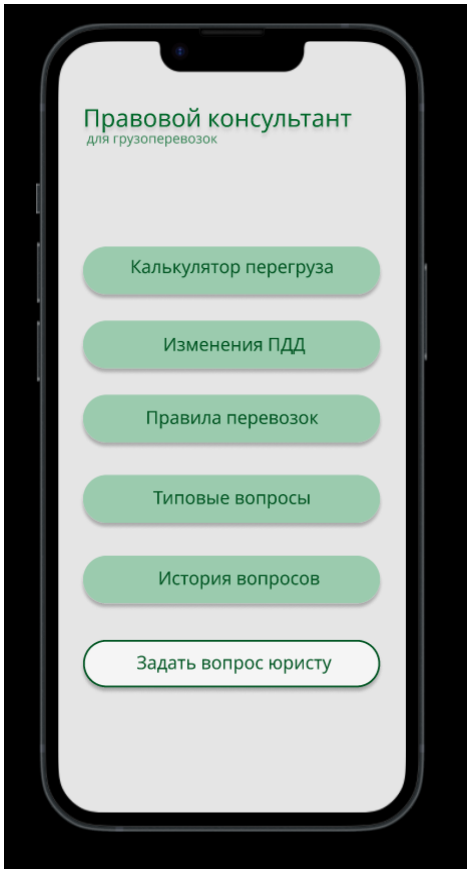


Рис. 1. Меню для зарегистрированного пользователя

историю своих обращений (рис. 2).

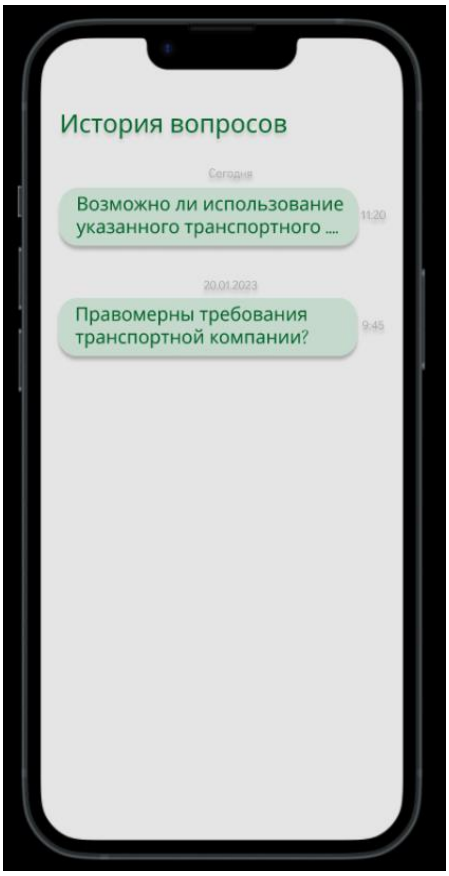


Рис. 2. История вопросов специалистам

В основе справочной системы лежит банк данных. В свою очередь банк данных делится на базы данных, формирующих весь информационный массив. Для формирования баз данных используются типовые подходы:

1. Ориентация на информационные потребности пользователей.
2. Группировка информации в соответствии с различными классификационными признаками, например: кодексы, конституционные законы, федеральное и региональное законодательство, официальные и неофициальные документы, отраслевое законодательство, нормы регламенты, стандарты; комментарии специалистов.

Обновление и пополнение информационного банка системы осуществляется на ежедневной основе за счет появления новых документов, а также формирования комментариев, ссылок, ответов на вопросы. Для обновления данных необходимо наличие технических возможностей и соответствующего пользовательского соглашения между клиентом и разработчиком информационной системы.

Заключение

Использование «Правового консультанта для грузоперевозчиков» позволит водителям оперативный ответ на постоянно возникающие правовые вопросы без существенных затрат времени на поиск необходимой информации. В качестве дальнейшего шага можно рассматривать создание аналогичного мобильного приложения.

Литература

1. Конституция Российской Федерации : принята всенародным голосованием 12 декабря 1993 года с изменениями, одобренными в ходе общероссийского голосования 01.07.2020. – Москва : Омега-Л, 2021. - 39 с.
2. Гражданский кодекс Российской Федерации (часть первая) от 30.11.1994 № 51-ФЗ (ред. от 29.12.2022) (с изм. и доп. от 06.08.2021) [Электронный ресурс] – URL: http://www.consultant.ru/document/cons_doc_LAW_5142/
3. Федеральный закон "Устав автомобильного транспорта и городского наземного электрического транспорта" от 08.11.2007 N 259-ФЗ (последняя редакция) [Электронный ресурс] – URL: https://www.consultant.ru/document/cons_doc_LAW_72388/
4. Духно Н. А. Транспортное право. Общая часть : учебник / отв. ред. Н. А. Духно, А. И. Землин. — М. : Юридический институт МИИТа, 2017. — 259 с.

УДК 519.8

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ СОВМЕСТНОГО ФИНАНСИРОВАНИЯ СОЦИАЛЬНО-ЗНАЧИМЫХ ПРОЕКТОВ РЕГИОНА

С.С. Полозова

Воронежский государственный университет

Введение

За последние годы область корпоративных финансов претерпела огромные изменения, результатом которых стало более ясное представление о способах формирования пакета социально-экономических проектов региона. Основной характеристикой современного управления региональным развитием является поиск направлений и механизмов взаимодействия органов государственного управления и инвесторов (представителей бизнеса) с целью совместного решения важных для региона социально-экономических задач. Активная мотивация и привлечение предпринимателей и частных инвесторов к решению актуальных проблем региона показали хороший результат в некоторых субъектах РФ, где показатели социально-экономические развития занимают наивысшие позиции в рейтингах [1].

Позитивный результат такого взаимодействия стал толчком для улучшения механизмов партнерства государства и бизнеса, его организационной, экономической, социальной, и других составляющих ([2] – [5]). Совместная реализация социально-значимых проектов региона является одной из форм государственно-частного партнерства. Кроме финансирования со стороны предпринимателя и инвестора, такие проекты предполагают привлечение государственных средств в форме субсидий. Финансовые вложения каждого участника такого трехстороннего соглашения обговариваются заблаговременно и четко фиксируются в контракте или контрактах. В этом случае актуальной теоретической и практической задачей является разработка инструментария формирования параметров контрактов – размеров первоначальных взносов и субсидий и распределение прибыли от проекта между инвестором и предпринимателем. При этом необходимым условием соглашения является положительная выгода от реализации проекта, которая для региональных органов власти заключается в социальном эффекте.

В настоящей статье рассмотрен подход к определению параметров контрактов корпоративного финансирования социально-значимых проектов, основанный на теоретико-игровых моделях. Первая из моделей позволяет рассчитать параметры контракта при корпоративном финансировании проекта средствами предпринимателя и инвестора без привлечения государственных субсидий. Вторая модель представляет собой двухуровневую схему заключения контрактов при участии региональных органов власти. На первом уровне модели определяется оптимальная величина субсидии, а на втором – параметры контракта предпринимателя и инвестора. Перейдем к рассмотрению предлагаемого подхода.

1. Материалы и методы

Рассмотрим проект, инициатором которого является некоторый предприниматель или группа предпринимателей (агент). Предполагаем, что проект является масштабируемым с постоянной нормой прибыли.

Будем считать, что реализация проекта требует начальных инвестиций в размере J, в то

время, как предприниматель может вложить в проект сумму $0 \le A_0 < J$. Недостающие средства в размере $J - A_0$ должны быть получены посредством привлечения внешнего инвестора.

Полагаем, что реализуемый проект имеет не только экономическую значимость с позиции получения прибыли предпринимателя и инвестора, но и социальную значимость для того региона, на территории которого он реализуется. Например, использует продукцию региональных производителей, обеспечивает создание новых рабочих мест, рост номинальной заработной платой, способствует повышению качества жизни населения и т.п. При этом полагаем, что совокупный внешний (социальный) эффект пропорционален размеру инвестиций $E = E(J)$, где $E(J)$ – строго возрастающая функция, частным случаем которой является линейная зависимость $E(J) = E_0 \cdot J$, $E_0 > 0$ – коэффициент пропорциональности. Совокупный внешний эффект может измеряться в безразмерных единицах или же в денежных единицах в форме явных доходов в бюджет региона.

В силу социальной значимости проекта региональные органы власти могут предоставить субсидию на его реализацию в размере S_0.

В настоящей работе будем предполагать, что внешние эффекты не зависят от успешности реализации проекта и стороны, реализующие проект (предприниматель и инвестор) руководствуются собственными экономическими интересами.

Для простоты изложения подхода рассматриваем статический случай, полагая, что проект длится один временной период, начинается в момент времени $t_0 = 0$ и заканчивается в момент $t_1 = 1$.

Вероятность того, что проект будет успешным ($NPV > 0$) обозначим через p. Соответственно, $1 - p$ – вероятность того, что проект не будет успешным. Внешний (социальный) эффект проекта не зависит от его успешности. В то время, как в случае успеха проект приносит доход, пропорциональный вложениям: $\alpha \cdot J$, где $\alpha \ge 1$.

Таким образом, общий доход от реализации проекта составляет $\alpha \cdot J + E(J)$ в случае его успешной реализации, и $E(J)$ в противном случае.

Следуя предположению работы [6], будем считать, что предприниматель может некачественно выполнять свою работу по реализации проекта или же преследовать собственную выгоду (например, закупать ресурсы по завышенным ценам в обмен на личные блага, вовремя не выполнять обязательства или тратить значительное время на оформление документов и т.п.). В этом случае (когда усилия предпринимателя - низкие) вероятность успешности проекта $p = p_l$. Если же предприниматель затрачивает на реализацию проекта значительные (высокие) усилия, то вероятность $p = p_h$, где $p_h > p_l$.

Частная выгода, которую предприниматель получает при недобросовестном поведении (низких усилиях) зависит от начальной стоимости проекта и составляет $B \cdot J$, где $B > 0$ – коэффициент пропорциональности.

Совместная реализация проекта предпринимателя и инвестора предполагает заключения контракта. Предприниматель разрабатывает контракт, который инвестор может принять или не принять. Если контракт принимается обеими сторонами, проект реализуется.

В предлагаемом инвестору контракте указывается:

1) начальная стоимость проекта (*J*);

2) финансовые средства, которые получат предприниматель и инвестор в случае

успешной реализации проекта (R_e и R_i соответственно).

При этом полагаем, что в случае неуспешной реализации проекта, и предприниматель, и инвестор, не получают прибыли.

Рассмотрим два случая:

1) региональные органы власти не поддерживают проект субсидиями;

2) региональные органы власти заинтересованы в реализации проекта и для его поддержки могут предоставить субсидии.

В первом случае при разработке контракта предприниматель максимизирует свою чистую ожидаемую выгоду:

$$p_h \cdot \alpha \cdot J - p_h R_i - A_0 \to \max. \quad (1)$$

При этом контракт должен гарантировать, что предприниматель будет затрачивать высокие усилия на реализацию проекта, иначе NPV будет отрицательным, и инвестор откажется в финансировании проекта:

$$p_h R_e \geq p_l R_e + B \cdot J. \quad (2)$$

Кроме того, инвестор должен быть заинтересован в реализации проекта т.е. ожидаемый доход от проекта должен быть не ниже инвестиций:

$$p_h \cdot R_i \geq J - A_0. \quad (3)$$

При этом полученный положительный доход должен быть распределен между предпринимателем и инвестором:

$$R_e + R_i = \alpha \cdot J. \quad (4)$$

Ограничения на переменные модели:

$$I \geq 0, \quad (5)$$

$$R_e, R_i \geq 0. \quad (6)$$

Задача определения оптимальных параметров контракта при корпоративном финансировании проекта (1)-(6) представляет собой задачу линейного программирования, переменными в которой являются I, R_e, R_i. Рассмотрим ее практическую реализацию.

2. Практическая реализация

Выражая из равенства (4) переменную $R_i = \alpha \cdot J - R_e$, задачу (1)-(6) можно свести к эквивалентной задаче линейного программирования следующего вида:

$$p_h R_e - A_0 \to \max, \quad (7)$$

$$B \cdot J - \left(p_h - p_l\right) R_e \leq 0, \quad (8)$$

$$\left(1 - \alpha \cdot p_h\right) J + p_h R_e \leq A_0, \quad (9)$$

$$-\alpha \cdot J + R_e \leq 0, \ (-R_e \leq 0) \quad (10)$$

$$J, R_e \geq 0. \quad (11)$$

Удобство задачи (7)-(11) заключается в том, что она содержит две переменные и может быть решена графически или с использованием симплекс-метода.

В предположении, параметры задачи таковы, что оптимальная точка лежит на пересечении прямых, определяемых соотношениями (8) и (9), рассчитаем их координаты.

Обозначим $\Delta p = p_h - p_l$ и выразим из соотношения (8) $J = R_e \frac{\Delta p}{B}$ и подставим в (9):

$$R_e \frac{\Delta p}{B}(1-\alpha p_h) + p_h R_l = A_0 \qquad (12)$$

Упростив (12), выражаем формулы для следующих переменных:

$$\left(R_e\right)^* = \frac{A_0}{\frac{\Delta p}{B} - p_h\left(\alpha \cdot \frac{\Delta p}{B} - 1\right)}, \left(J\right)^* = \frac{\Delta p \cdot A_0}{\Delta p - p_h\left(\alpha \cdot \Delta p - B\right)}, \qquad (13)$$

$$\left(R_i\right)^* = \alpha \cdot J - \frac{A_0}{\frac{\Delta p}{B} - p_h\left(\alpha \cdot \frac{\Delta p}{B} - 1\right)}. \qquad (14)$$

Рассмотрим второй случай, когда региональные органы власти заинтересованы в реализации проекта и могут предоставить субсидии на его поддержку.

В этом случае полагаем, что разработкой контракта занимаются региональные органы власти. Предприниматель может согласиться с условиями контракта или отклонить его. В случае согласия с контрактом, предприниматель предлагает на рассмотрение контракт инвестору. Инвестор может согласиться с условием контракта предпринимателя или отклонить его. В таком случае ставится задача отыскания оптимальных параметров пары контрактов: «государство-предприниматель» (контракт верхнего уровня) и «предприниматель-инвестор» (контракт нижнего уровня).

Согласно контракту верхнего уровня государство в лице региональных органов власти в начале проекта предоставляет субсидию в размере $\overline{S}$, которая не подлежит возврату. При этом региональные органы власти заинтересованы в максимизация общей социальной полезности, определяемой как сумма частной и государственной выгоды от реализации проекта за вычетом стоимости проекта:

$$U = p \cdot \alpha \cdot J + E_0 \cdot J - I = \left(p \cdot \alpha \cdot J - pR_i - A\right) + \left(E_0 \cdot J - \overline{S}\right).$$

В этом случае задача определения оптимального размера субсидии в контакте верхнего уровня имеет следующий вид:

$$U\left(\overline{S}\right) = \left(p \cdot \alpha \cdot J - pR_i - A_0\right) + \left(E_0 \cdot J - \overline{S}\right) \to \max. \qquad (15)$$

При условии:

$$E_0 \cdot J - \overline{S} \ge 0, \quad \overline{S} \ge 0. \qquad (16)$$

При этом J и R_i являются оптимальными параметрами контракта нижнего уровня и находятся при каждом значении $\overline{S}$ как решение следующей задачи:

$$p_h R_e - A_0 \to \max, \qquad (17)$$

$$B \cdot J - \left(p_h - p_l\right) R_e \le 0, \qquad (18)$$

$$\left(1 - \alpha \cdot p_h\right) J + p_h R_e \le A_0 + \overline{S}, \qquad (19)$$

$$-\alpha \cdot J + R_e \le 0, \qquad (20)$$

$$J, R_e \ge 0. \qquad (21)$$

Задача (15)-(16) представляет собой двухуровневую иерархическую игру. Для отыскания ее решения аналогично (13)-(14) выпишем аналитическое представление решения задачи нижнего уровня как функции от $\overline{S}$:

$$R_e = R_e\left(\bar{S}\right) = \frac{A_0 + \bar{S}}{\frac{\Delta p}{B} - p_h\left(\alpha \cdot \frac{\Delta p}{B} - 1\right)}, \tag{22}$$

$$J = J\left(\bar{S}\right) = \frac{\Delta p \cdot \left(A_0 + \bar{S}\right)}{\Delta p - p_h\left(\alpha \cdot \Delta p - B\right)}, \tag{23}$$

$$R_i = R_i\left(\bar{S}\right) = \alpha \cdot J - \frac{A_0 + \bar{S}}{\frac{\Delta p}{B} - p_h\left(\alpha \cdot \frac{\Delta p}{B} - 1\right)}. \tag{24}$$

Подставим (22)-(24) в задачу определения оптимального размера субсидии в контракте верхнего уровня (15)-(16):

$$U\left(\bar{S}\right) = \left(p \cdot \left(\alpha \cdot \frac{\Delta p \cdot \left(A_0 + \bar{S}\right)}{\Delta p - p_h\left(\alpha \cdot \Delta p - B\right)} + \frac{A_0 + \bar{S}}{\frac{\Delta p}{B} - p_h\left(\alpha \cdot \frac{\Delta p}{B} - 1\right)} \right) - A_0 \right) +$$

$$+ \left(E_0 \frac{\Delta p \cdot \left(A_0 + \bar{S}\right)}{\Delta p - p_h\left(\alpha \cdot \Delta p - B\right)} \cdot - \bar{S} \right) \to \max.$$

$$E_0 \cdot \frac{\Delta p \cdot \left(A_0 + \bar{S}\right)}{\Delta p - p_h\left(\alpha \cdot \Delta p - B\right)} - \bar{S} \geq 0, \quad \bar{S} \geq 0. \tag{25}$$

Пусть $\left(\bar{S}\right)^*$ – решение задачи (25). Подстановка $\left(\bar{S}\right)^*$ в формулы (22)-(24) позволяет найти оптимальные значения параметров контракта не только верхнего, но и нижнего уровня.

Заключение

Предложенный в настоящей работе подход к определению параметров контрактов корпоративного финансирования социально-значимых проектов включает две теоретико-игровые модели. Теоретической основой моделей выступает теория контрактов. Отличительной особенностью подхода является учет возможности привлечения государственных субсидий, в связи с чем возникает необходимость заключения контрактов двух уровней - верхнего (государство и предприниматель) и нижнего (предприниматель и инвестор). Моделью такого взаимодействия является иерархическая игра. В статье получено аналитическое решение каждой из моделей при некоторых предположениях, что позволяет применить подход в практике управления.

Литература

1. Сборник тезисов заявок Всероссийского конкурса «Лучшие управленческие решения региональных органов власти по развитию инвестиционной среды» 2014 г. — Москва: Издательство «МаркетМаш Принт», 2014. — 228 с.

2. Бондаренко Ю.В. Математический подход к определению финансовой поддержки социально значимых проектов муниципального образования / Ю.В.Бондаренко, А.Н.Чикомазов // Экономика и менеджмент систем управления, 2016. – Т. 21.– № 3.2. – С. 204-212.

3. Bondarenko Yu. V. The task of coordinating of economic indicators of the development of the region and the mathematical approach to its solution / Yu. V. Bondarenko, I.V. Goroshko, I.L. Kashirina // Journal of Physics: Conference Series, 2019. – Vol. 1203.– P. 012037 (doi:10.1088/1742-6596/1203/1/012037).

4. Barkalov S.A. Designing systems of group stimulation in the management of energy complex objects / S.A. Barkalov, V.N. Burkov, P.N. Kurochka // Advances in Intelligent Systems and Computing, 2019. – Vol. 983.– P. 55-68.

5. Дабагян Е.К. Развитие государственно-частного партнерства в Российской Федерации / Е.К. Дабагян // Российское предпринимательство, 2015. – Т. 16. – № 4 (274). – С. 611-622.

6. Tirole J. The theory of corporate finance / J. Tirole.– Princeton : Princeton University Press.– 645 p.

УДК (519.1)

МОДЕЛИРОВАНИЕ ЗАДАЧИ МАРШРУТИЗАЦИИ ТРАНСПОРТА С ДВУМЯ ВИДАМИ ОБЪЕКТОВ И ЧЕРЕДОВАНИЕМ

С. А. Полянская, Н. С. Сафонов

Воронежский государственный университет
Университетская пл., 1, 394018 Воронеж, Российская Федерация

Аннотация. В статье рассматривается постановка задачи маршрутизации транспорта с несколькими центрами с чередованием. Также рассмотрены различные алгоритмические подходы к решению данной задачи. Важным аспектом построения маршрутов является обязательное чередование объектов разных типов. Осмысленную постановку можно представить как задачу по сбору стогов сена в грузовики в поле. Для решения этой задачи были рассмотрены жадный и муравьиный алгоритмы, а также точный алгоритм, основанный на методе ветвей и границ. Приводятся результаты вычислительного эксперимента. В заключении представлены основные результаты работы.

Ключевые слова: задача маршрутизации, жадный алгоритм, муравьиный алгоритм, метод ветвей и границ, эвристические методы.

Введение

Задача планирования оптимальных маршрутов является актуальной в сфере логистики. Одной из основных целей оптимизации составления маршрутов является максимизация прибыли, и минимизация расходов, связанных с затратами на перемещения. Существует ряд алгоритмов, решающих поставленную задачу. В данной работе будет исследованы и описаны некоторые из них.

1. Постановка задачи маршрутизации транспорта

В данной главе представлено формальное описание условия задачи маршрутизации транспорта, а также разработка и разбор математических моделей.

Рассмотрим следующую задачу: существует совокупность неподвижных объектов двух типов: целевые объекты (тип A) и центры (тип B). Известны затраты, необходимые для перемещения между объектами. Необходимо посетить все целевые объекты таким образом, чтобы суммарные затраты были минимальными. Причём целевые объекты и центры должны чередоваться в маршруте, при этом идея чередования будет включена в введённых переменных. Кроме того, каждый целевой объект в совокупности можно посетить только один раз, а любой центр можно посетить сколько угодно раз.

Под затратами подразумевается расстояние, стоимость, временные затраты и т.п.

На рис. 1 приведён пример решения задачи маршрутизации транспорта с двумя видами объектов.

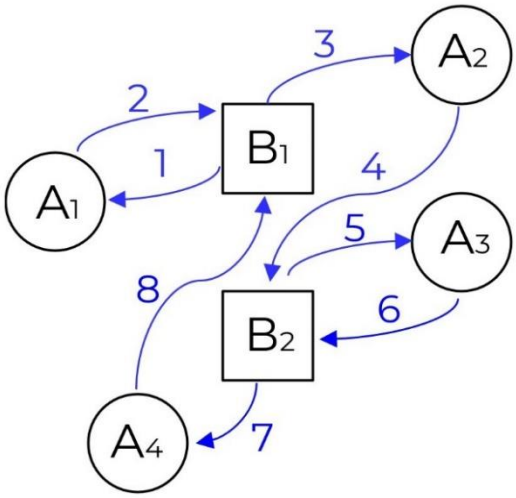


Рис. 1. □ *– центр,* ○ *– целевой объект*
Маршрут мобильного объекта

2. Краткий обзор существующих решений

В данной главе будут рассмотрены основные методы и алгоритмы решения задачи маршрутизации транспорта, а также приведены их достоинства и недостатки.

Метод ветвей и границ [1] — алгоритмический метод для нахождения оптимальных решений различных задач оптимизации. Метод является развитием метода полного перебора, в отличие от последнего — с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений.

Ключевое преимущество метода ветвей и границ заключается в том, что он является точным алгоритмом, то есть позволяет найти гарантированно верное решение.

Жадный алгоритм [2] — алгоритм, заключающийся в принятии локально оптимальных решений на каждом этапе. Данный алгоритм не всегда приводит к оптимальному решению задачи, так как принятие оптимальных решений на первых этапах алгоритма приводит к тому, что на последних этапах может быть выбрано не оптимальное решение.

Главным преимуществом алгоритма является его простота и скорость работы. Для определения локальных решений не нужны дополнительные вычисления различных коэффициентов или параметров. Также поиск решения с помощью жадного алгоритма занимает немного времени из-за отсутствия анализа принимаемых решений.

Муравьиный алгоритм [3] — это приближенный алгоритм, использующийся для решения различных сложных оптимизационных задач. Идея этого алгоритма основана на моделировании поведения муравьиной колонии, выполняющей поиск пути от муравейника к источнику пищи.

Для ряда случаев муравьиные алгоритмы позволяют находить весьма оптимальные решения, однако имеют высокую трудоемкость при больших размерностях задачи по сравнению с более точными методами, а также содержат управляющие параметры, для которых требуется подбирать значения для каждого нового набора входных данных.

3. Математическая модель задачи маршрутизации транспорта с двумя типами объектов и чередованием

Пусть имеется m центров (объектов типа B) и n целевых объектов (объектов типа A).

(c_{ji}) — матрица затрат на перемещение между *j*-ым целевым объектом и *i*-ым центром;

Рассмотрим также некоторые ограничения, возникновение которых будет следовать из условия задачи:

1. Общая длина маршрутов должна быть минимальной.
2. В каждый целевой объект приезжает ровно один мобильный объект.
3. Из каждого целевого объекта уезжает ровно один мобильный объект.
4. Если мобильный объект приезжает в целевой объект, то он должен из него уехать.
5. Если мобильный объект уезжает из целевого объекта, то он должен был в него приехать.
6. Количество въездов в каждый центр равно количеству выездов из него.
7. Маршрут, составленный для каждого мобильного объекта, должен быть связным. Это ограничение представляет собой условие отсутствия подциклов, аналогичное тому, что представлено в задаче коммивояжера [4].

3.1. Модель «целевой объект – центр – целевой объект»

Построим математическую модель для перемещения мобильного объекта вида «целевой объект – центр – целевой объект».

Введем следующие переменные:

$x_{jil} = \{0,1\}$, где $x_{jil} = 1,$ если мобильный объект перемещается из *j*-го целевого объекта к *i*-му центру, и затем в *l*-ый целевой объект; $x_{jil} = 0,$ $j = 1,...,n,$ $i = 1,...,m,$ $l = 1,...,n.$

Матрица с суммарными затратами для последующего поиска минимума суммы будет вычисляться следующим образом:

$c_{jil} = c_{ji} + c_{il},$ $j = 1,...,n,$ $i = 1,...,m,$ $l = 1,...,n.$

Заметим, что это верно в том случае, если затраты симметричны, в противном случае нужно использовать элементы c_{ji} и c_{il}, что обозначают несимметричные затраты из целевого объекта *j* в центр *i* и затраты из центра *i* в целевой объект *l*.

3.2. Модель «центр – целевой объект – центр»

Построим математическую модель для перемещения вида «центр – целевой объект – центр».

Введем следующие переменные:

$x_{ijk} = \{0,1\}$, где $x_{ijk} = 1$, если мобильный объект перемещается из *i*-го центра к *j*-му целевому объекту, и затем в *k*-ый центр; $x_{ijk} = 0,$ иначе; $i = 1,...,m,$ $j = 1,...,n,$ $k = 1,...,m.$

Матрица с суммарными затратами будет вычисляться следующим образом:

$c_{ijk} = c_{ij} + c_{jk},$ $i = 1,...,m,$ $j = 1,...,n,$ $k = 1,...,m.$

Заметим, что это верно в том случае, если затраты симметричны, в противном случае нужно использовать элементы c_{ij} и c_{jk}.

3.3. Сравнение двух моделей с точки зрения ограничений

Рассмотрим ограничения для математической модели 3.1 «целевой объект – центр – целевой объект»:

1. Общая длина маршрутов должна быть минимальной. Целевая функция, в таком случае, будет иметь вид:

$$\sum_{j=1}^{n}\sum_{i=1}^{m}\sum_{l=1}^{n} c_{jil}x_{jil} \to \min \quad (1)$$

2. В каждый целевой объект приезжает ровно один мобильный объект.

Ограничение 2 можно выразить следующей формулой:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{jil} = 1, \quad l = 1,\ldots,n. \quad (2)$$

3. Из каждого целевого объекта уезжает ровно один мобильный объект.

Ограничение 3 можно выразить следующей формулой:

$$\sum_{i=1}^{m}\sum_{l=1}^{n} x_{jil} = 1, \ j = 1,\ldots,n. \quad (3)$$

Выполнение ограничений 4 и 5 автоматически следует из формул (2) и (3).

Ограничение 6 выполняется за счёт введённых переменных.

Далее рассмотрим ограничения для математической модели 3.2 «центр – целевой объект – центр»:

1. Общая длина маршрутов должна быть минимальной. Целевая функция, в таком случае, будет иметь вид:

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m} c_{ijk}x_{ijk} \to \min, \quad (4)$$

2. В каждый целевой объект приезжает ровно один мобильный объект.

3. Из каждого целевого объекта уезжает ровно один мобильный объект.

Ограничения 2 и 3 можно выразить следующей формулой:

$$\sum_{i=1}^{m}\sum_{k=1}^{m} x_{ijk} = 1, \quad j = 1,\ldots,n, \quad (5)$$

Ограничения 4 и 5 выполняются за счёт введённых переменных.

6. Количество въездов в каждый центр равно количеству выездов из него.

Ограничение 6 можно выразить следующим образом:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{kji,} \quad k = 1,\ldots,m, \quad (6)$$

Проанализировав ограничения для двух математических моделей, можно сделать вывод, что они отличаются мини-маршрутом, инициализированным при построении математических моделей, а также вычислением целевой функции и ограничениями на въезд и выезд, различия которых исходят из введённых переменных.

4. Описание методов решения

Рассмотрим решение данной задачи с использованием простого жадного алгоритма, алгоритма муравьиной колонии, а также метода ветвей и границ. Далее будет приведено описание разработанных алгоритмов для двух моделей.

4.1. Жадный алгоритм для модели «центр – целевой объект – центр»

Алгоритм 1. Описание жадного алгоритма для математической модели 3.1.

1. Ввести номер i_init начального центра (объект типа B).

2. Задать множество $J = \{1, 2, ..., n\}$ всех целевых объектов и минимальную длину пути $min\_path = 0$.

3. Для каждого текущего центра $i\_curr = \overline{1,m}$ найти ближайший доступный целевой объект $j \in J$, для которого, в свою очередь, найти ближайший центр $k = \overline{1,m}$, получая минимальный суммарный путь между этими тремя объектами:

$$c\_min = \min_{j \in J} \left(\min_{k=1..m} \left(c_{i\_curr, j, k} \right) \right).$$

4. Вычислить длину минимального пути на данном этапе:

$$min\_path = min\_path + c\_min.$$

5. Задать значение переменной:

$$x_{i\_curr, j, k} = 1.$$

6. Исключить выбранный целевой объект j (объект типа A) из множества всех рассматриваемых целевых объектов:

$$J = J / \{ j \}.$$

7. Повторять шаги 3 – 6 пока не будут посещены все целевые объекты, т.е. пока $J \neq \varnothing$.

8. Если все объекты посещены, возвращаемся в начальный:

$$x_{i\_curr, j, i\_init} = 1.$$

Результатом выполнения алгоритма будет матрица $\left(x_{ijk} \right)_{m \times n \times m}$, задающая найденный замкнутый маршрут, а также значение результирующего (минимального) суммарного пути min_path.

Заметим, что на шаге 6 выполняется условие посещения целевого объекта только один раз. На шаге 8 выполняется ограничение, отвечающее условию того, что количество въездов и выездов для каждого объекта типа B равны.

4.2. Жадный алгоритм для модели «целевой объект – центр – целевой объект»

Алгоритм 2. Описание жадного алгоритма для математической модели 3.2.

1. Ввести номер j_init начального целевого объекта (объект типа A).

2. Задать множество $J = \{1, 2, ..., n\}$ всех целевых объектов и минимальную длину пути $min\_path = 0$.

3. Для каждого текущего целевого объекта $j\_curr = \overline{1,n}$ найти ближайший центр $i = \overline{1,m}$, для которого, в свою очередь, найти ближайший доступный целевой объект $l \in J$, получая минимальный суммарный путь между этими тремя объектами:

$$c\_min = \min_{i=1..m} \left(\min_{j \in J} \left(c_{j\_curr, i, l} \right) \right).$$

4. Вычислить длину минимального пути на данном этапе:

$$min\_path = min\_path + c\_min.$$

5. Задать значение переменной:

$$x_{j\_curr, i, l} = 1.$$

6. Исключить выбранный целевой объект j (объект типа А) из множества всех рассматриваемых целевых объектов:

$$J = J \setminus \{j\}.$$

7. Повторять шаги 3 – 6 пока не будут посещены все целевые объекты, т.е. пока $J \neq \varnothing$.
8. Если все объекты посещены, возвращаемся в начальный:

$$x_{j\_curr,i,j\_init} = 1.$$

Результатом выполнения алгоритма будет матрица $\left(x_{jil}\right)_{n\times m\times n}$, задающая найденный замкнутый маршрут, а также значение результирующего (минимального) суммарного пути min_path.

Заметим, что на шаге 6 выполняется условие посещения целевого объекта только один раз. На шаге 8 выполняется ограничение, отвечающее условию того, что количество въездов и выездов для каждого объекта типа В равны.

4.3. Алгоритм муравьиной колонии для модели «целевой объект – центр – целевой объект»

Для предложенного муравьиного алгоритма, процесс поиска следующего стога (целевого объекта) и грузовика (центра) будет реализован с учетом вероятностей выбора того или иного пути. Таким образом, важным этапом алгоритма является вероятностный переход, который заключается в необходимости перехода в одну из нескольких точек на основе имеющихся вероятностей перехода.

Алгоритм 3. Описание алгоритма вероятностного выбора.

1. Задать множество M всех объектов и соответствующие и вероятности p_i, причем $\sum_i p_i = 1$.
2. Сгенерировать $rand \in [0,1]$ – границы вероятностного выбора.
3. Ввести $sum_p = 0$ – значение суммы вероятностей.
4. Ввести $i = 1$ – порядковый номер объекта в множестве M.
5. Положить $sum_p = sum_p + p_i$ – увеличение суммарной вероятности.
6. Если $sum_p < rand$, то положить $i = i + 1$ и перейти к шагу 5, иначе – конец алгоритма.

Результатом выполнения алгоритма будет индекс i – порядковый номер объекта в множестве M.

Алгоритм 4. Описание алгоритма муравьиной колонии для математической модели 3.1.

1. Задать:

$amount_{it}$ – число итераций алгоритма;

$amount_{ant}$ – число муравьев в колонии;

$\tau_{jil} = 1$ – значение следа феромона на мини-маршруте « j - й целевой объект $\rightarrow i$ - й центр $\rightarrow l$ - й целевой объект»;

$d_{jil} = \dfrac{1}{c_{jil}}$ – желательность мини-маршрута « j - й целевой объект $\rightarrow i$ - й центр $\rightarrow l$ - й целевой

объект»;

α – коэффициент влияния феромона на вероятность выбора пути;

β – коэффициент влияния желательности на вероятность выбора пути;

ρ – коэффициент испарения феромона;

$min\_path^* = \infty$ – результирующая длина минимального пути (рекорд).

2. Положить:

$it = 1$ – номер текущей итерации алгоритма;

$ant = 1$ – порядковый номер муравья в колонии.

3. Обозначить:

$I = \{1, 2, ..., m\}$ – множество всех центров;

$J = \{1, 2, ..., n\}$ – множество всех целевых объектов;

$min\_path = 0$ – длина минимального пути.

4. Рассчитать вероятность выбора мини-маршрута «j - й целевой объект $\rightarrow i$ - й центр $\rightarrow l$ - й целевой объект»:

$$p_{jil} = \frac{(\tau_{jil})^{\alpha} * (d_{jil})^{\beta}}{\sum_i \sum_l (\tau_{jil})^{\alpha} * (d_{jil})^{\beta}}.$$

5. Применить алгоритм 3, где положить $M = I$, получить индекс i^*.

6. Применить алгоритм 3, где положить $M = J$, получить индекс l^*.

7. Вычислить длину минимального пути на данном этапе:

$$min\_path = min\_path + c_{ji^*l^*}.$$

8. Задать значение переменной:

$$x_{ji^*l^*} = 1.$$

9. Исключить выбранные целевые объекты j и l^* из множества всех рассматриваемых целевых объектов:

$$J = J \setminus \{j, l^*\}.$$

10. Если $min\_path < min\_path^*$, то запомнить рекорд длины минимального пути:

$$min\_path^* = min\_path.$$

11. Если $x_{jil} = 1$ – мини-маршрут «j - й целевой объект $\rightarrow i$ - й центр $\rightarrow l$ - й целевой объект» пройден, то изменить значение следа феромона:

$$\tau_{jil} = \tau_{jil} * (1 - \rho) + d_{jil},$$

иначе:

$$\tau_{jil} = \tau_{jil} * (1 - \rho).$$

12. Если $ant < amount_{ant}$, то увеличить порядковый номер муравья в колонии и перейти к шагу 3:

$$ant = ant + 1.$$

13. Если $it < amount_{it}$, то увеличить номер текущей итерации алгоритма, перейти к шагу 2:

$$it = it + 1,$$

иначе – конец алгоритма.

Результатом выполнения алгоритма будет матрица $\left(x_{jil}\right)_{n \times m \times n}$, задающая найденный замкнутый маршрут, а также значение результирующего (минимального) суммарного пути min_path^*.

4.4 Алгоритм метода ветвей и границ для модели «целевой объект – центр – целевой объект»

Для предложенного метода ветвей и границ, процесс поиска следующего целевого объекта будет реализован с учётом того, что всё множество Ω можно разделить на области и искать решение на этих частях. При этом явно не перспективные части можно будет отбросить.

Алгоритм 5. Общий алгоритм метода ветвей и границ

1. Инициализация

Задать начальное значение рекорда R. Если нахождение начального значения затруднительно, то $R = +\infty$, $I = \varnothing$ - множество номеров подмножеств, подлежащих ветвлению.

$J = \{0\}$ – множество номеров подмножеств, для которых будет решаться оценочная задача.

2. Вычисление оценок

Решить оценочные задачи для множества Ω_j, $j \in J$. Вычислить значение оценок $\xi(\Omega_j)$.

3. Обновление рекорда

Если на шаге 2 получены допустимые точки x^p, $p = \overline{1, P}$, то положить в качестве рекорда:

$R = (R, \min f(x^p)), p = \overline{1, P}$

4. Сокращение перебора

Осуществить закрытие не перспективных множеств, включая даже тех, на которых оценка равна рекорду. Удалить их номера из множеств I, J. $I = I \cup J$, $J = \varnothing$ - И если $I = \varnothing$, то переход к шагу 7.

5. Реализация стратегии

В соответствии с выбранной стратегией из множества номеров I выбрать номер k множества Ω_k, который подлежит ветвлению.

6. Ветвление

Осуществить разбиение множества Ω_k на непересекающиеся подмножества так, что $\Omega_k = \bigcup_i \Omega_{k_i}$ и положить $J = J \cup \{k_i\}$. Перейти к шагу 2.

7. Конец алгоритма

$f_{\min} = R$. $f_{\min}$ полагается равным последнему рекорду.

Алгоритм 6. Обход по минимальным оценкам

1. Определить начальный рекорд

В частности, когда у нас нет информации в качестве начального рекорда можно взять ∞, или длину любого маршрута. Здесь же приводим исходную матрицу. Вычисляем оценку первоначального множества и положим $k = 0$.

2. Выбрать пару (p, q)

3. Провести ветвление $\Omega_k = \Omega_{k_1} + \Omega_{k_2}$

4. Вычисление матриц C_{k_1} и C_{k_2}

Если какая-то из этих матриц имеет размерность 2x2, то переход к шагу 7.

5. Вычисление оценок $\xi(\Omega_{k_1}) = \xi(\Omega_{k_2})$

6. Выбор перспективного множества решения Ω_s в соответствии стратегии обхода. Положить $k = s$ и переход к шагу 2.

7. Получение допустимого маршрута. Возможна смена рекорда и сокращения перебора.

8. Проверка критерия оптимальности. Если выполнен – останов, иначе переход к шагу 6.

5. Вычислительный эксперимент

Далее будет представлен результат вычислительного эксперимента разработанных алгоритмов 1 и 2. В качестве входных данных использовалась выборка из пятидесяти матриц (C_{ijk}) и (C_{jil}), затраты которых для перемещения от центра до целевого объекта и от целевого объекта до центра, заданы случайным образом в диапазоне от 1 до 10.

Результаты вычислительного эксперимента представлены в табл. 1. В приведенной таблице, в первом столбце указана размерность задачи: количество центров m и целевых объектов n. Второй и третий столбцы отвечают за среднее значение целевой функции (длину минимального пути) $\min\_path$, вычисленной для моделей «центр – целевой объект – центр» и «целевой объект – центр – целевой объект» соответственно.

Таблица 1

Результаты вычислительного эксперимента

Размерность задачи	**Алгоритм 1**	**Алгоритм 2**
$m=2,\ n=10$	*min_path* = 37	*min_path* = 19
$m=2,\ n=20$	*min_path* = 68	*min_path* = 32
$m=2,\ n=50$	*min_path* = 154	*min_path* = 63
$m=2,\ n=100$	*min_path* = 270	*min_path* = 114
$m=5,\ n=15$	*min_path* = 25	*min_path* = 24
$m=5,\ n=35$	*min_path* = 51	*min_path* = 49
$m=5,\ n=50$	*min_path* = 73	*min_path* = 65
$m=5,\ n=100$	*min_path* = 138	*min_path* = 115
$m=7,\ m=30$	*min_path* = 40	*min_path* = 43
$m=8, n=50$	*min_path* = 66	*min_path* = 62
$m=8, n=100$	*min_path* = 124	*min_path* = 114
$m=10, n=20$	*min_path* = 31	*min_path* = 29
$m=10, n=50$	*min_path* = 69	*min_path* = 63
$m=10, n=100$	*min_path* = 114	*min_path* = 113

Выводы вычислительного эксперимента

Проанализировав результаты вычислительного эксперимента, можно отметить, что алгоритм 1 для модели «центр – целевой объект – центр» показывает, что среднее значение

целевой функции уменьшается, когда число целевых объектов уменьшается по сравнению с количеством центров. Алгоритм 2 для модели «целевой объект – центр – целевой объект» имеет результаты вычисления длины минимального пути более выгодные, в сравнении с вычислениями алгоритмом 1 при одинаковых размерностях задачи.

Заключение

В данной работе было проведено исследование алгоритмов решения задачи маршрутизации с двумя типами объектов и чередованием, а также были выполнены все следующие задачи:

1. Рассмотрены две математические модели для задачи маршрутизации с двумя типами объектов и чередованием;
2. Изучен метод решения этой задачи с помощью жадного алгоритма;
3. Изучен метод решения этой задачи с помощью муравьиного алгоритма;
4. Изучен метод решения этой задачи с помощью метода ветвей и границ;
5. Реализованы два алгоритма для двух из рассмотренных математических моделей поставленной задачи;
6. Проведён вычислительный эксперимент и его анализ.

Доцент, канд. физ.-мат. наук, доцент кафедры вычислительной математики и прикладных информационных технологий ВГУ, Медведев Сергей Николаевич.

Литература

1. Медведев, С. Н. Математическая модель и алгоритм решения задачи маршрутизации транспортных средств с несколькими центрами с чередованием и единым местом сбора // Вестник ВГУ, Серия: Системный анализ и информационные технологии 2021, № 1. – 2021. DOI: https://doi.org/10.17308/sait.2021.1/3368.
2. Пожидаев, М. С. Алгоритмы решения задачи маршрутизации транспорта: автореф дис. канд. техн. наук. – Томск: ТГУ, 2010. – 16 с.
3. Dorigo, M. Ant colony system: A cooperative learning approach to the traveling salesman problem [Текст] / Dorigo M., Gambardella L.M.// IEEE Transactions on Evolutionary Computation 1, 1997.
4. Корбут, А. А. Дискретное программирование / А. А. Корбут, Ю. Ю. Финкельштейн [под ред. Д. Б. Юдина]. – Москва: Наука, 1969. – 368 с.

УДК: *519.6*

СИМВОЛЬНЫЕ (АНАЛИТИЧЕСКИЕ) ВЫЧИСЛЕНИЯ В СИСТЕМАХ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ: СРАВНИТЕЛЬНЫЙ АНАЛИЗ

М. А. Поминов

Воронежский государственный университет

Введение

Использование больших объёмов данных, которые характеризуют современные научные, производственные, информационные процессы, требует всё более активного внедрения специализированных информационных систем. Особое место среди них занимают системы компьютерной математики, использующие в своей работе символьные (аналитические) вычисления.

С помощью алгоритмов программирования можно решать задачи в таких областях как химия (в том числе, фармацевтика), физика, роботостроение, космическая индустрия, развитие технологий искусственного интеллекта и т.д.

Основным отличием СКМ (систем компьютерной математики) является оперирование не только числовыми, но и строковыми (символьными) данными, что значительно ускоряет проведение расчётов, делает их более структурированными и позволяет задействовать путём унификации данные самых различных форматов.

Существует ряд систем компьютерной математики – Maple, Mathematica, MathCad, MuPAD, Macsyma, Reduce, Axiom, Magma и другие – использующиеся для решения универсальных и специфических задач. В данной статье рассматриваются особенности четырёх наиболее востребованных в российском научном сообществе систем и даётся их табличный сравнительный анализ.

1. Системы компьютерной математики

Далее, опуская подробное описание понятия аналитических вычислений и принципа работы СКМ, рассматриваются конкретные аналитические системы и приводится их сравнительный анализ. В число наиболее известных систем компьютерной математики входят: Maple, Mathematica, Mathcad, Matlab.

1.1. Maple

Одна из старейших систем компьютерной математики (разрабатывалась с начала 1980-х гг.), активно применяющаяся в академической и практической научной среде. Символьный анализатор данной СКМ используется также в системах более поздних поколений, таких как Mathcad и Matlab.

Визуально работа в Maple схожа с работой в текстовом редакторе, в котором используются такие объекты как: строки ввода и вывода, комментарии, графики, гиперссылки (как элемент внутренней навигации). Структура программного модуля включает в себя секции, подсекции разного уровня, базовый элемент – ячейка.

Основным техническим документом, с которым работает оператор Maple, является лист, который может выступать как в роли среды для решения поставленной вычислительной или

аналитической задачи, так и в роли пространства подготовки технической документации по проекту.

1.2. Mathematica

Данная система компьютерной математики была разработана в конце 1980-х гг., новейшая версия вышла в конце 2022 года. Продолжается активная разработка и совершенствование этой среды. Отличается расширенными редакторскими и функциональными возможностями, позволяющими осуществлять работу не только в режиме текстового редактора, но и в интегрированных средах разработки.

В процессе анализа, вычисления и преобразования данных могут использоваться не только числовые и символьные объекты, но также графический и видеоформат. Наличие мультипользовательского интерфейса позволяет осуществлять групповую работу надо поставленной задачей вычислительного или аналитического характера.

Ещё одна особенность – синхронизация процесса вычисления с внесением изменений в исходные данные. Так, если значение одного из использующихся в вычислении параметров было обновлено, то алгоритм вычисления итогового результата автоматически перезапустится и данные обновятся на всех его этапах. Это позволяет избежать неверной интерпретации корреляции между входными и выходными параметрами вычисления.

1.3. Mathcad

В отличие от двух предыдущих СКМ, данная система ориентирована не столько на академические научные исследования, сколько на практическое использование при решении конкретных задач в таких областях как проектирование машин и аппаратов, зданий и сооружений, инженерных систем и компьютерного программного обеспечения.

Для применения Mathcad конечный пользователь не должен владеть полной информацией о параметрах функций, задействованных в вычислительных и аналитических процессах, таких, как методы и алгоритмы вычисления. Ещё одной особенностью является то, что для ввода символьных параметров можно использовать графический метод – специальную кнопочную панель интерфейса программной среды.

В Mathcad есть возможность создания обучающих документов для коллег или студентов, что позволяет эффективно использовать систему не только в рабочем, но и в образовательном процессе.

1.4. Mathlab

Эта система компьютерной алгебры также используется, в первую очередь, в практических целях: инженерное проектирование, алгоритмизация вычислительных процессов, сложные технические и научные вычисления.

Отличительная особенности системы состоит в ориентации на матричную интерпретацию данных, так называемое «векторное мышление», а также визуализацию данных как в стандартном графическом, так и в анимированном формате. Matlab способен проводить параллельную запись всех вычислительных операций в процессе их осуществления, что позволяет, в случае получения неверного или неожиданного результата поэтапно проанализировать цепочку вычислений и обнаружить ошибку или точку поворота.

Matlab – среда, интерфейс которой легко настроить под конкретных пользователей и задачи, а также его можно создать с нуля и оптимизировать под привычную конечным пользователям рабочую среду. При этом внутренние программы и алгоритмы генерируются

сравнительно просто и выглядят компактно и эргономично.

2. Сравнение системы компьютерной математики

В качестве основных параметров для сравнения перечисленных выше СКМ взяты такие параметры, как сфера практического применения, поддерживаемые операционные системы и языки программирования, возможность проводить как числовые, так и символьные вычисления, функциональность и недостатки каждой из систем.

Следует особо отметить, что декларируемая универсальность практически каждой из рассматриваемых системы компьютерной математики не может рассматриваться как реальная. Так, Maple и Mathematica больше ориентированы на использование при проведении фундаментальных научных исследований, а Mathcad и Matlab – на применение в практических сферах деятельности, от разработки ПО до промышленного производства и строительства.

Отличия систем компьютерной математики

	Maple	Mathematica
Сфера применения	Академическая наука, сфера образования, инженерное проектирование	Академическая наука, сфера образования, инженерное проектирование
Операционные системы и приборы	Персональные компьютеры, рабочие станции Sun, специализированные ЭВМ; Windows, Linux, Unix.	Персональные компьютеры; Windows, Windows Server
Язык программирования	4GL, совместим с C, C#, Fortran, LaText и др.	Wolfram, совместим с C, C++, Fortran, Excel, LabView
Поддержка численных вычислений	Да	Да
Поддержка символьных вычислений	Да	Да
Функциональность	Вычислительные задачи, линейная алгебра, дифференциальное вычисление, интегральное вычисление, создание двухмерных и трёхмерных графиков и графических функций, тензорная алгебра, евклидова геометрия, аналитическая геометрия, элементы высшей математики (теория	Вычислительные задачи, интегральное вычисление, дифференциальное вычисление, вейвлет-анализ, преобразование рядов и функций, нахождение пределов, поддержка двухмерной и трёхмерной графической информации, работа с материалами в формате аудиовизуального потокового канала.

	вероятностей, комбинаторика и т.д.). Возможна внешняя интеграция машинного обучения на языке Python.	Работает в режиме мультизадачности. Существует ручной и автоматический режим машинного обучения.
Недостатки	Не очень высокая скорость работы, однозадачность (во время вычислительного процесса редактирование документов невозможно), высокая стоимость лицензии.	Специализированный язык отличается периодическим возникновением ошибок и сбоев, в частности, в компиляторе. Низкая скорость работы.

	Mathcad	Matlab
Сфера применения	Практическая химия, физика, машиностроение, электроника, гражданское строительство, образовательная сфера, инженерия	Академическая наука, программирование, математическое моделирование, инженерное проектирование, разработка приложений, визуализация, образовательная сфера
Операционные системы и приборы	Персональные компьютеры; Windows, Linux, Java	Персональные компьютеры; Windows, Linux, macOS, Solaris.
Язык программирования	C++	Matlab, совместим с C, C++, Fortran, Java
Поддержка численных вычислений	Да	Да
Поддержка символьных вычислений	Да	Да
Функциональность	Вычислительные задачи, статистические расчёты, дифференциальное вычисление, построение двухмерных и трёхмерных графиков, матричное вычисление, многоступенчатое	Вычислительные задачи, матрицы, интерполяция, линейная алгебра, статистический анализ, дифференциальное вычисление, алгоритмизация, визуализация данных.

	программирование, работа с различными системами единиц измерения, использование греческого алфавита в символьных вычислениях.	
Недостатки	Ориентация на практические задачи делает СКМ не очень удобной для научных целей, однако такое использование тоже возможно. Отсутствие открытого кода, невозможность пользовательских изменений. Недоступность ПО физическим лицам – только образовательным, научным и коммерческим организациям.	Узкая специализация и невозможность использования за пределами собственной платформы, высокая цена. В связи с ориентацией на графическую подачу данных – низкая скорость работы.

Заключение

Развитие символьной математики является логическим продолжением развития вычислительной математики, открывающее неограниченные возможности для анализа и разработки новых технологий, а также внедрения их во все сферы человеческой жизнедеятельности – от производственной до культурно-досуговой. Использование систем компьютерной математики, оперирующих символьными объектами анализа, позволяет реализовывать новые технические решения без привлечения специалистов-аналитиков, путём создания алгоритма с заданными параметрами.

Современный выбор систем компьютерной алгебры позволяет подобрать наиболее подходящую из них для решения конкретной теоретической или практической задачи. Также видится логичным возможность развития имеющихся и создание новых аналитических систем, которые, в том числе, будут направлены на решение специфических отраслевых задач и позволят осуществить скачок в научном развитии информационно-емких сфер знания и деятельности человека.

Литература

1. Косарев, В. И. 12 лекций по вычислительной математике / В. И. Косарев ; – Москва : Физматкнига, 2013. – 240 с.

2. Рябенький, В. С. Введение в вычислительную математику / В. С. Рябенький ; – Москва : Физматлит, 2000. – 296 с.

3. Дьяконов, В. П. Энциклопедия компьютерной алгебры / В. П. Дьяконов ; – Москва : ДМК Пресс, 2009. – 1257 с.

4. Жидков, Е. Н. Вычислительная математика: Учебное пособие / Е. Н. Жидков ; – Москва : Академия, 2013. – 208 с.

УДК *004.043*

МЕТОДЫ АНАЛИЗА И ВЫЯВЛЕНИЯ ЗАКОНОМЕРНОСТЕЙ НА ПРИМЕРЕ МЕДИЦИНСКИХ ДАННЫХ

Д. В. Попов

Воронежский государственный университет

Введение

Анализ больших данных (Big Data) — это процесс сбора, хранения, обработки больших объемов информации. С развитием технологий, обрабатываемые объемы данных увеличиваются, и это приводит к потребностям использования методов машинного обучения и искусственного интеллекта для решения многих задач [1]. Одной из таких задач является анализ данных в медицине, который позволяет выявлять тенденции в заболеваниях и прогнозировать вероятность их возникновения.

Рассмотрим возможности анализа больших данных в медицине с целью выявления тенденций в заболеваниях и прогнозирования вероятности их возникновения. Основная задача заключается в определении наиболее эффективных методов сбора, обработки и анализа больших объемов данных, а также в установлении возможных ограничений и проблем, связанных с использованием данных методов. Кроме того, необходимо определить потенциальные ценности анализа больших данных в медицине и возможности использования его результатов для улучшения диагностики, профилактики и лечения заболеваний.

1. Актуальность задачи

Анализ больших данных в медицине позволяет проводить исследования, направленные на выявление причин и факторов развития заболеваний, а также оценивать эффективность методов лечения. При этом с помощью алгоритмов машинного обучения и статистических методов анализируются огромные объемы данных, например, данные об истории болезни пациентов, результаты лабораторных исследований, данные о генетическом коде и другие.

В частности, анализ данных может использоваться для выявления тенденций в заболеваниях. Например, анализируя данные о пациентах, можно выявить наиболее распространенные заболевания, а также факторы, способствующие их возникновению.

Кроме того, анализ данных позволяет прогнозировать вероятность возникновения заболеваний. Используя данные об истории болезни и другие параметры (например, возраст, наследственность, образ жизни и т.д.), можно построить модель, которая предскажет вероятность возникновения заболевания в будущем. Это позволяет проводить более точный и персонализированный подход к лечению и профилактике заболеваний.

Выявление тенденции в заболеваниях и прогнозирование вероятности их возникновения может существенно улучшить качество оказания медицинской помощи, а также сократить расходы на здравоохранение. В связи с этим, разработка новых методов сбора, хранения и обработки больших объемов информации в медицине является актуальной и востребованной задачей.

2. Методы сбора данных

Существует множество методов сбора медицинских данных, которые могут быть

использованы для анализа больших объемов информации в медицине. Некоторые из наиболее распространенных включают в себя:

1. Сенсорные устройства: электрокардиограммы, измерители пульса, кислорода в крови и другие медицинские датчики, которые позволяют собирать данные о физиологических функциях пациента.

2. Медицинские информационные системы (МИС): электронные медицинские карты, истории болезни, результаты анализов, лекарственные назначения и другие данные, которые содержатся в МИС, могут быть использованы для анализа здоровья пациентов.

3. Геномные базы данных: сбор и хранение данных о геномах пациентов, что позволяет проводить более точный анализ заболеваний.

4. Исследования на основе опросов: анкеты, опросы и другие формы сбора данных, которые могут использоваться для определения факторов риска и причин возникновения заболеваний.

5. Биомаркеры: сбор и анализ данных о биомаркерах, таких как метаболиты, белки и гормоны, которые могут быть использованы для диагностики и прогнозирования заболеваний.

6. Социальные медиа: данные, собранные из социальных сетей и форумов, могут быть использованы для анализа мнения и опыта пациентов, а также для выявления тенденций и паттернов в заболеваниях.

Каждый из этих методов имеет свои преимущества и ограничения, и выбор метода зависит от целей и задач анализа.

Собранные медицинские данные должны храниться в соответствии с требованиями к конфиденциальности и защите персональных данных. Облачные службы являются современным и удобным способом хранения данных. Они позволяют доступ к информации из любой точки мира и обеспечивает высокий уровень безопасности. Кроме того, облачное хранилище данных обычно обладает высокими скоростями передачи данных и большим объемом хранения, что делает его идеальным для нашей задачи. Однако, использование облачного хранилища данных требует высокой степени доверия к поставщику услуг и может стать дорогостоящим. Также, наличие Интернет-соединения является обязательным, что может быть недоступно в некоторых районах.

Примером такой облачной платформы служит HDInsight. Она позволяет развертывать кластеры Hadoop, Spark, Hive, Storm, Kafka и других технологий для обработки и анализа больших объемов данных.

3. Обработка данных

3.1. Статистический анализ

Статистический анализ данных — это метод обработки медицинских данных, который используется для выявления статистически значимых связей между различными факторами и заболеваниями. Данный способ может включать в себя такие статистические методы, как расчет стандартных отклонений, корреляций, анализ дисперсии (ANOVA), линейную и логистическую регрессию и другие.

Основным преимуществом статистического анализа данных является его широкое применение в медицинских исследованиях, а также простота интерпретации результатов. Этот метод может помочь выявить значимые связи между факторами и заболеваниями, такие как связь между курением и раком легких, между возрастом и сердечно-сосудистыми заболеваниями и другими [2].

Однако, у этого метода есть и ограничения. Например, статистический анализ данных может быть недостаточно точным для выявления сложных взаимосвязей между факторами и заболеваниями, особенно если эти связи не являются прямолинейными. Также, при

использовании этого метода необходимо учитывать возможность смещения данных и влияния других факторов, которые могут оказывать влияние на результаты.

Тем не менее, статистический анализ данных все еще является важным инструментом для обработки медицинских данных и выявления связей между факторами и заболеваниями, особенно в случаях, когда необходимо быстро получить результаты и сделать предварительные выводы.

3.2. Машинное обучение

Машинное обучение — это метод обработки данных, который основан на использовании алгоритмов и моделей, которые обучаются на основе исходных данных и могут предсказывать результаты на новых данных.

Одним из наиболее распространенных методов машинного обучения является метод классификации. Этот метод используется для классификации данных на основе заранее определенных категорий. Например, в медицине можно использовать метод классификации для определения, имеет ли пациент определенное заболевание или нет.

Другим методом машинного обучения является метод кластеризации, который используется для группировки данных на основе сходства между ними. Например, можно использовать метод кластеризации для группировки пациентов на основе сходства их медицинских данных.

Одним из преимуществ методов машинного обучения является их способность работать с большими объемами данных и выявлять сложные взаимосвязи между факторами и заболеваниями. Кроме того, эти методы могут работать с неструктурированными данными, такими как изображения и тексты, что делает их полезными для анализа медицинских данных.

Однако, у методов машинного обучения есть и некоторые ограничения. Например, они могут требовать больших вычислительных мощностей и больших объемов данных для обучения моделей. Кроме того, необходимо учитывать возможность переобучения моделей, когда модель слишком хорошо подстраивается под исходные данные и не может корректно работать с новыми данными.

3.3. Data Mining

Метод Data Mining является процессом автоматического поиска скрытых, ранее неизвестных, но потенциально полезных закономерностей в больших объемах данных. Он применяется для обнаружения неочевидных связей и закономерностей, которые могут помочь в понимании данных и принятии бизнес-решений.

Метод Data Mining включает в себя широкий спектр техник, таких как кластерный анализ, классификация, ассоциативные правила и другие, и он может использоваться для анализа различных типов данных, включая текстовые, числовые и многомерные данные [3].

Применение Data Mining в медицине может помочь в выявлении скрытых закономерностей в заболеваниях, выделении групп риска, прогнозировании течения заболеваний и принятии решений по лечению. Например, при анализе медицинских записей можно выделить группы пациентов с определенными заболеваниями и факторами риска, что может помочь в принятии решений по лечению и профилактике.

Однако, метод Data Mining также имеет свои ограничения. Например, для успешного применения этого метода требуется большое количество данных, что может быть проблематично в некоторых медицинских областях. Кроме того, результаты Data Mining могут быть не совсем точными, если использованные данные были не полными или некорректными.

4. Проблемы и ограничения

Хотя анализ больших данных имеет большие перспективы в области здравоохранения, существуют также серьезные проблемы и ограничения. Одной из самых больших проблем является конфиденциальность и безопасность данных. Медицинские данные являются конфиденциальными, и существуют строгие правила, обеспечивающие конфиденциальность пациентов. Медицинские организации должны обеспечить, чтобы их методы хранения и анализа данных соответствовали этим правилам.

Кроме того, качество данных может быть существенным ограничением, поскольку анализ больших данных опирается на высококачественные и точные данные для получения значимой информации. В большинстве случаев, эти данные хранятся в разных форматах и системах, что затрудняет их объединение и использование для анализа.

Заключение

Анализ больших данных является мощным инструментом для выявления тенденций в заболеваниях и прогнозирования вероятности их возникновения. Результаты исследований показывают, что с помощью различных методов обработки данных, таких как машинное обучение, статический анализ, Data Mining можно получить ценную информацию о заболеваниях и оценить риски их возникновения у пациентов.

Однако, при использовании больших объемов медицинских данных необходимо учитывать проблемы конфиденциальности, этические и правовые вопросы, а также возможные ошибки в данных. Также необходимо разработать более эффективные алгоритмы обработки, учитывающие специфику медицинских данных.

Тем не менее, все больше исследований показывают, что анализ больших данных является важным инструментом в медицине и может привести к более точному диагностированию и прогнозированию заболеваний, что в конечном итоге поможет улучшить качество жизни пациентов и снизить затраты на здравоохранение.

Литература

1. Ухлова, В. В. Основы технологий Big Data / В. В. Ухлова // Воронеж: Издательский дом ВГУ, 2020. – 81 с

2. Кэти О’Нил Убийственные большие данные. Как математика превратилась в оружие массового поражения / Кэти О’Нил // Москва: Издательство АСТ, 2018. – 320 с.

3. Data mining технология. – Режим доступа: https://www.ibm.com/topics/data-mining (дата обращения: 05.04.2023)

УДК 004.032, 331.108

ВОЗМОЖНОСТИ ПРИМЕНЕНИЯ НЕЙРОННОЙ СЕТИ ПРИ ОРГАНИЗАЦИИ ПРОЦЕССА ПОДБОРА ПЕРСОНАЛА В СТАРТАП-ПРОЕКТАХ

А. Э. Потемкина

Воронежский государственный университет

Введение

Целью данной статьи является демонстрация возможностей нейронных сетей в процессе подбора кадров. Рассматривается процесс подбора кадров для старап-проектов. Нейронные сети рассматриваются как инструмент прогнозной аналитики. Мониторинг и заполнение профессиональных пробелов в организациях могут быть реализованы с помощью технологий анализа данных. В рамках данной работы задача ограничена поиском кадров для стартап-проектов.

2. Проблемы стартап-проектов

С позиции управления стартап рассматривается как высокорисковый проект. Источниками рисков являются как сами процессы реализации стартапа, так и команда. Проблемы стартапа по стадиям жизненного цикла представлены на рис. 1.

Рис. 1. Проблемы стартапа по стадиям жизненного цикла

Одной из проблем стартапа, которая имеет место быть на протяжении всего жизненного цикла, является проблема команды. Изучение методик формирования команд для различного типа проектов, в том числе стартапов, показывает, что единого подхода к формированию эффективных команд не существует. Одни методики основываются на личностном подходе, другие на компетенциях, покрывающих должностные обязанности, т.е. важно подбирать и

специалистов высокого уровня, и личностей, способных достигать в кратчайшие сроки результатов. При этом, каждый из кандидатов должен уметь работать в команде. Более того, важно не только сформировать команду, но сохранить ее и сделать эффективной на протяжении всего жизненного цикла проекта.

Рассмотрим процесс формирования команды с позиции методологии Института управления проектами, представленной на рис. 2. Если посмотреть жизненный цикл стартапа, то каждый из этапов можно детализировать до задач, решаемых при управлении проектами, поскольку значительная часть деятельности связана непосредственно с разработкой цифрового сервиса. И на каждой стадии управления проектом на каждом этапе жизненного цикла стартапа список компетенций участников команды становится все более расширенным, при этом он связан с различными сферами деятельности.

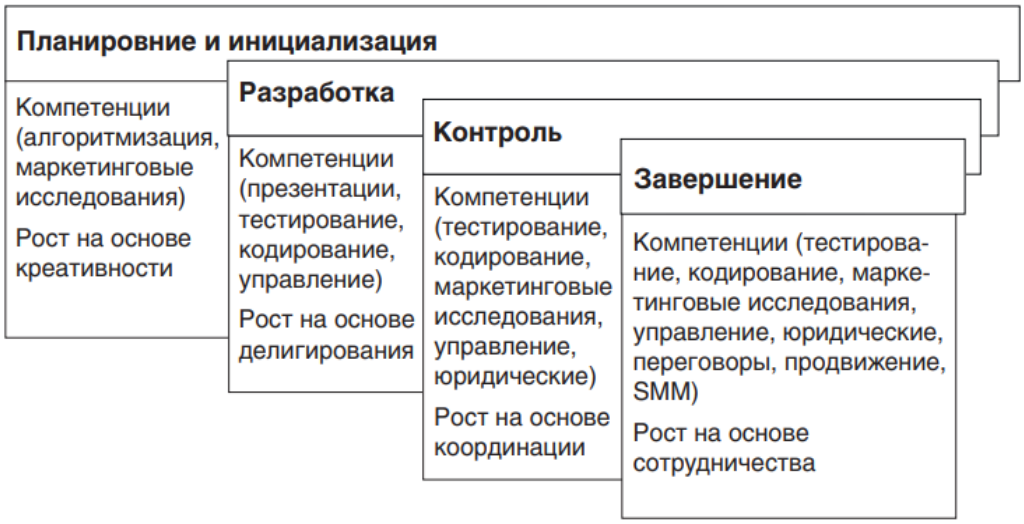


Рис. 2. Проблемы управления проекта в методологии Института управления проектами

Как следствие, учесть все компетенции потенциального члена команды на ранних стадиях проекта невозможно. Анализ существующих методик показывает, что прогнозной модели для подбора участников команды, учитывающих компетенции по стадиям жизненного цикла нет, в то время как такой подход существенно снизил бы риски в управлении персоналом по проекту. Это и делает задачу актуальной.

2. Предлагаемое решение

2.1. Алгоритм решения

В основу предлагаемого решения положены онтологическая и нейросетевая модели. Применение онтологий позволит связать задачу оценки компетенций через преобразование запроса в набор терминов и понятий с конкретными потребностями проекта, планируемой задачи или выполняемой функции. Набор компетенций позволит достаточно точно описать трудовое поведение, которое требуется для успешного выполнения работы в данном проекте.

Разработка решения состоит из следующих этапов, представленных на блок-схеме на рис. 3. Результат поиска не всегда может быть релевантным запросам проектных задач и ожиданиям руководителя проекта. Во-первых, не все кандидаты, доступные анализу, могут соответствовать предъявляемым требованиям. Во-вторых, поиск осуществляется в ограниченном информационном пространстве и только по опубликованному предложению. Кроме того, при предварительном анализе могут быть отсеяны потенциально более приоритетные кандидаты. Последнее связано с субъективностью оценки или

несогласованности принимаемых решений лица, занимающегося кадровым подбором, и непосредственно менеджера проекта, для которого выполняется поиск персонала.

Рис. 3 Блок-схема разработки решения

2.2. Описание атрибутов и сущностей решения

Хранимые взаимосвязанные параметры и атрибуты персонала представляют собой исторические данные. Данные и функционал образуют единую самообучаемую нейронную сеть на базе онтологии данных параметров и атрибутов. Основные сущности решения: сотрудники, структурные подразделения компании, проекты и дополнительная информация.

Дополнительная информация может включать в себя сертификаты, обучение, участие в мероприятиях. Для каждой из сущностей предполагается наличие набора описательных параметров и атрибутов, которые могут изменяться во времени.

Атрибутами разрабатываемой модели данных являются:

- роли;

- компетенции;
- критерии;
- оценки.

Значение данных атрибутов с течением времени может изменяться, поэтому стоит хранить историю изменений по данным параметрам. Набор аналитических показателей, рассчитывается на основании информации о сотрудниках, выставляемых по ним оценкам, изменении их навыков и прочих атрибутов. Критерии оценивают по балльной шкале. В каждый критерий могут входить требования, которые повышают баллы. Примером матрицы может служить вариант, представленный в таблице 1. Матрица компетенций и полномочий по проектным ролям включает в себя аналитика (А), внедренеца (В), инженера сопровождения (И), менеджера клиентских проектов (М), разработчика (Р), руководителя (РЛ). Выделенные компетенции представлены в табл. 1.

Таблица 1

Матрица компетенций и полномочий

Компетенции/роли	А	В	И	М	Р	РЛ
Владение базовыми навыками	+	+	+	-	+	-
Постоянное расширение компетенций	-	+	-	+	-	-
Знание процесса выпуска версий продуктов	-	-	+	-	+	+
Знание процессов аналитики	+	-	-	-	-	-
Знание стандартов пользовательского интерфейса	+	+	+	-	+	-
Умение выявлять ошибки логики и неоптимальные пути в алгоритмах	+	-	+	-	-	-
Управление ожиданиями клиента	-	-	+	+	-	-
Разрешение конфликтных ситуации на проекте	-	-	+	+	-	+
Планировка задач в масштабах проекта	-	-	-	+	-	+
Участие в принятии решения о концепциях	+	+	+	+	+	+
Разрешение спорных ситуаций с заказчиком/клиентом	-	+	+	+	-	+
Общение с заказчиком/клиентом	-	-	+	+	-	+

По рассчитанным критериям и выявленным способностям для каждой вакантной проектной роли возможно простроить аналитику, сравнить способности каждого из претендентов, выстроить рейтинг специалистов по рассчитанным критериям и зарплатным ожиданиям.

Нейронная сеть позволит учесть связи различных атрибутов с принимаемыми решениями в зависимости от значений рассчитанных критериев. Входными параметрами для модели нейронной сети предлагается выбрать: возраст, набор из характеристик пользователя, число сертификатов, город, информацию о проектах, уровень заработной платы, ВУЗ и навыки. Результатом применения прогнозной модели на основе нейронной сети будет расчет критериев по каждой проектной роли.

Заключение

Предполагается, что использование нейронной сети позволит упростить подбор персонала в стартап-проекты, позволяющий сформировать эффективные команды. Результаты расчетов могут быть также представлены в виде диаграммы рассеяния, которая отобразит распределение кандидатов на разные должности в зависимости от их входных и выходных

параметров. Результаты работы могут быть рекомендованы для департаментов подбора персонала компаний в сфере информационных технологий.

Литература

1. Азарнова, Т. В. Применение методов интеллектуального анализа данных в оценке функциональной эффективности команд менеджеров / Т. В. Азарнова, И. М. Терлюга, В. В. Ухлова // Вестник Воронежского государственного университета. Сер. Системный анализ и информационные технологии .— Воронеж, 2020 .— №4. – С. 50-61.
2. Арефьев, А.О. Управление компетенцией и ротация человеческих ресурсов проектно-ориентированного предприятия / А.О.Арефьев, А.Д. Баженов // Internationa! Journal of Management Science and Engineering Management. - 2008. - №. 3. - С. 163-175.
3. Белбин, Р.М. Команды менеджеров. Секреты успеха и причины неудач / Пер. с англ. Москва : Юрайт, 2013. – 411 с.
4. Жуков, Ю. М. Технологии командообразования / Ю. М. Жуков, А. В. Журавлёв, Е. Н. Павлова // учеб. пособие для студ. вузов . – М.: Аспект Пресс, 2008. – 320 с.
5. Каширина, И. Л. Нейросетевые и гибридные системы : учебно-методическое пособие для вузов / И.Л. Каширина, Т.В. Азарнова ; Воронеж. гос. ун-т .— Электрон. текстовые дан. — Воронеж : Издательский дом ВГУ, 2014. – 289 с.
6. Сайт по поиску и подбору персонала. – Режим доступа: http://www.hh.ru. – (Дата обращения: 16.03.2023).
7. Ухлова, В. В. Применение системного подхода в процедурах формирования команд стартапов ИТ-проектов Актуальные проблемы прикладной математики, информатики и механики / В. В. Ухлова, С. А. Палкина // Сборник трудов Международной научной конференции, Воронеж, 13-15 декабря 2021 г. — Воронеж, 2022. — С. 791-796

УДК 004.75

СТРУКТУРА ЦИФРОВОЙ ЭКОСИСТЕМЫ ДЛЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

М. А. Принев

Воронежский государственный университет

Введение

Современный уровень развития цифровых технологий подразумевает многообразие программных и аппаратных решений и необходимость их продуктивного взаимодействия. В процессе разработки программного обеспечения зачастую возникают проблемы, связанные с усложнением структуры и повышенными требованиями к производительности используемых устройств.

Для оптимизации работы программных и аппаратных продуктов, используемых при разработке программного обеспечения, необходимо применять интеграционные решения, позволяющие снизить уровень нагрузки на информационные системы. Одним из используемых интеграционных решений является, так называемый, «экологический подход», когда программное обеспечение рассматривается, как часть цифровой экосистемы [1].

1. Аспекты структуры цифровой экосистемы

Цифровая экосистема представляет собой бесшовную открытую, адаптивную, распределенную систему, обладающую свойствами самоорганизации, масштабируемости и устойчивости. Принцип устройства такой системы заключается в том, что все программные модули должны работать в виде выделенных подсистем, взаимодействующих друг с другом, тем самым обеспечивая функционирование программного обеспечения в случае некоррект-ной работы или падения отдельных модулей. Таким образом, структурирование цифровой экосистемы является очень важным аспектом успешной работы.

Рассмотрим структуру цифровой экосистемы, представленную на рис. 1. В ее основе находятся три блока: собственные сервисы, сторонние системы и интерфейсы. Объединяет их шина данных, которая передаёт информацию по всей экосистеме

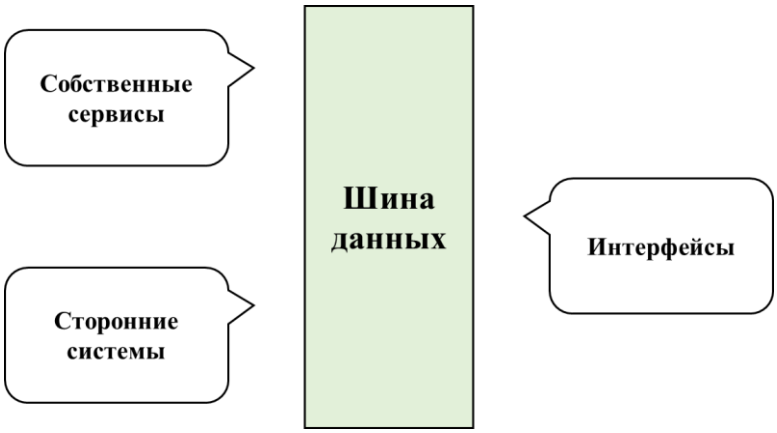


Рис. 1. Принципиальная схема структуры цифровой экосистемы

Собственные сервисы — это базис, на котором выстраивается весь проект. Они состоят из двух частей: ядра экосистемы и обслуживающего программного обеспечения. В ядре закладывается нужная функциональность системы и определяется, с какими данными она будет работать, настраивается функциональность экосистемы и прописываются процессы для автоматизации задач, уведомлений, триггеров на определённые действия и так далее.

В экосистеме мы имеем дело с сервисной архитектурой [2]. Все компоненты общаются не напрямую, а через шину данных. Использование шины данных дает преимущества перед прямым подключением систем друг к другу:

- При внесении изменений в подсистемы данные и связи не пострадают;
- Экосистема будет максимально защищена от сбоев, на время устранения неполадок функциональность подсистем будет ограничена, но не более того.

Посредством шины данных объединяются все модули экосистемы. Самые важные характеристики этого инструмента — автономность и устойчивость к высоким нагрузкам.

Сторонние системы представляют собой внешние сервисы и устройства, подключаемые для обеспечения функционала, заявленного в программном обеспечении. Интерфейс — это то, что видят все пользователи экосистемы. Сторонние системы и интерфейсы взаимодействуют с шиной по ключевым данным.

2. Цифровая экосистема SmartWall

Разработанный автором программно-аппаратный комплекс SmartWall [3, 4, 5], предназначенный для создания интерактивных неэлектронных поверхностей при помощи компьютерного зрения, имеет модульную структуру. Поэтому для минимизации вычислительных нагрузок и возможности использования маломощного оборудования целесообразной является разработка цифровой экосистемы с возможностью распределенных вычислений. Экосистема SmartWall будет представлять собой сеть, в которой будут объединены различные устройства, использующие ПО и обменивающиеся между собой информацией посредством информационной шины и специально разработанному протоколу, обеспечивающим защиту передаваемых данных. Устройства сети, которые в данный момент времени не решают задачи по взаимодействию с пользователем будут участвовать в системе распределенных вычислений. Также в экосистеме будут предусмотрены узловые точки, представляющие собой устройства, подключенные к сети интернет и осуществляющие связь с конечными локальными точками.

Структура экосистемы SmartWall будет содержать помимо основного ядра два кластера, являющихся по функциональности дополнительными ядрами, но имеющих специальный поток для передачи данных, что обеспечит дополнительный уровень распределения задач, а значит приведет к повышению быстродействия программных продуктов при использовании маломощных устройств, а также даст возможность использования локальных точек.

Результаты работы будут значительно повышать экономические показатели эффективности проекта, так как создание экосистемы приведет к улучшению технических параметров, увеличению качества и надежности конечного продукта без роста себестоимости.

Принципиальная схема структуры цифровой экосистемы SmartWall представлена на рис. 2.

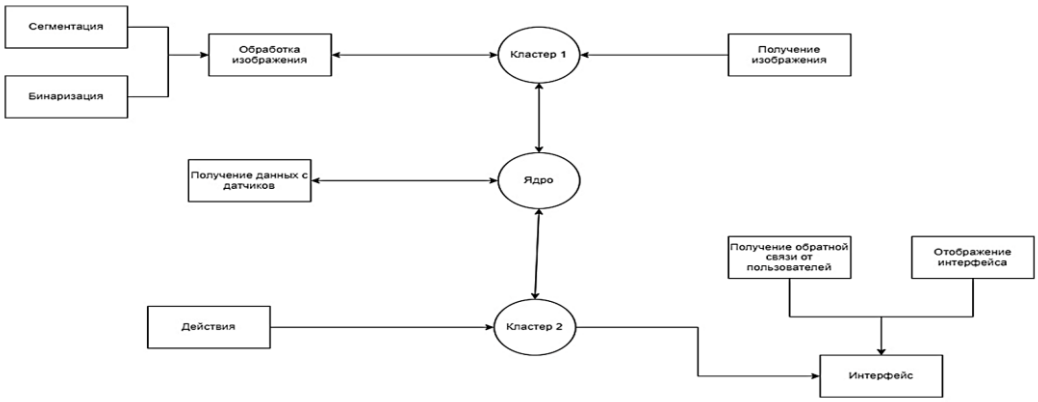


Рис. 2. Принципиальная схема структуры цифровой экосистемы SmartWall

Заключение

Цифровые экосистемы являются актуальным и трендовым техническим решением в условиях современности [6]. Разработка экосистемы SmartWall позволит не только сделать программно-аппаратный комплекс более доступным и привлекательным для пользователей, но и получить общие интеграционные решения для использования в более глобальных проектах.

Литература

1. Гопоненко А.А., Исхакова А.Р. ЦИФРОВЫЕ ЭКОСИСТЕМЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ // Международный студенческий научный вестник. – 2018. – № 5. . – Режим доступа: https://eduherald.ru/ru/article/view?id=18782 (Дата обращения: 11.04.2023).

2. Косяков М.С. Введение в распределенные вычисления. –СПб: НИУИТМО, 2014. – 155 с.

3. Свид. 2018611550 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. SmartWall / М. А. Принев; заявитель и правообладатель ФГБОУ ВО «Воронежский государственный университет» (RU). – №2017662835; заявл. 11.12.17; опубл. 02.02.18, Реестр программ для ЭВМ. – 1 с.

4. Свид. 2020618286 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Пакет SmartWall для различных операционных систем / М. А. Принев; заявитель и правообладатель Принев Мечислав Александрович (RU). – №2020617321; заявл. 02.07.20; опубл. 22.07.20, Реестр программ для ЭВМ. – 1 с.

5. Свид. 2020618747 Российская Федерация. Свидетельство об официальной регистрации программы для ЭВМ. Конструктор SmartWall / М. А. Принев; заявитель и правообладатель ФГБОУ ВО «Воронежский государственный университет» (RU). – №2020617762; заявл. 14.07.20; опубл. 04.08.20, Реестр программ для ЭВМ. – 1 с.

6. Концепция государственного регулирования цифровых платформ и экосистем – Режим доступа: https://www.economy.gov.ru/material/departments/d31/koncepciya_gos_regulirovaniya_cifrovyh_platform_i_ekosistem/ (Дата обращения: 11.04.2023).

УДК 004.738.5

ПРИНЦИПЫ РАЗРАБОТКИ ПРОГРЕССИВНЫХ ВЕБ ПРИЛОЖЕНИЙ

Н. В. Путинцев

Воронежский государственный университет

Аннотация. В работе проводится анализ прогрессивных веб-приложений (PWA) и рассматривается их влияние на пользовательский опыт и производительность веб-приложений. Представлены основные принципы создания PWA с помощью Service Worker'ов и Web App Manifest'ов, а также проведен анализ и оценка применения PWA на практике.
В статье рассмотрены как общие подходы к решению проблемы, так и конкретные примеры создания PWA с помощью элементов кода на JavaScript. **Ключевые слова**: прогрессивные веб-приложения, PWA, Service Worker, Web App Manifest, кэширование, offline-режим, JavaScript.

Введение

Современные веб-приложения предоставляют пользователю широкий спектр функциональных возможностей, которые ранее были доступны только на десктопных приложениях. Однако, с увеличением сложности веб-приложений и ростом количества пользователей, возникают проблемы с производительностью и масштабируемостью. Для решения этих проблем были разработаны прогрессивные веб-приложения, которые позволяют создавать более быстрые и масштабируемые приложения, используя новые технологии и архитектуры.

Одной из ключевых задач в разработке прогрессивных веб-приложений является анализ и выбор наиболее подходящих технологий и подходов. В данной статье проведен анализ различных методов реализации уведомлений в веб-приложениях, основанных на языке JavaScript. Рассмотрены основные подходы и библиотеки, такие как WebSocket, Server-Sent Events (SSE), COMET и long polling. Проведена оценка плюсов и минусов каждого из подходов, даны рекомендации по использованию в конкретных сценариях. Представленная информация полезна для разработчиков и архитекторов веб-приложений, которые хотят улучшить производительность и масштабируемость своих приложений с помощью прогрессивных технологий.

1. WebSocket

WebSocket — это протокол двусторонней связи между клиентом и сервером, который позволяет установить постоянное соединение между ними и обмениваться данными в режиме реального времени.

WebSocket может использоваться в PWA для реализации функциональности, которая обновляется в режиме реального времени, например, для получения уведомлений, обновления информации о статусе приложения и т. д.

При использовании WebSocket в PWA можно отправлять сообщения с клиента на сервер и получать обновления от сервера без необходимости повторного запроса каждый раз. Это

помогает существенно снизить нагрузку на сеть и ускорить обмен данными между клиентом и сервером.

Для использования WebSocket в PWA—возможно использовать стандартный API WebSocket в браузере. Пример кода, демонстрирующий, как установить соединение WebSocket и отправить сообщение с клиента на сервер:

```
const socket = new WebSocket('ws://localhost:8080');
socket.addEventListener('open', (event) => {
console.log('WebSocket connection opened');
socket.send('Hello, server!');
});
socket.addEventListener('message', (event) => {
console.log(`Received message from server: ${event.data}`);
});
socket.addEventListener('close', (event) => {
console.log('WebSocket connection closed');
});
```

В этом примере создается новый объект WebSocket и ему передается URL-адрес сервера. Затем добавляются обработчики событий для открытия соединения, получения сообщений от сервера и закрытия соединения.

Когда соединение WebSocket открывается,—отправляется сообщение на сервер, используя метод send. Когда сервер отправит ответное сообщение, его получение происходит в обработчике события message.

Таким образом, использование WebSocket в прогрессивных веб-приложениях позволяет обеспечить более быстрое и эффективное взаимодействие между клиентом и сервером, а также расширить функциональность приложения.

2. Server-Sent Events (SSE)

Server-Sent Events (SSE) — это один из способов реализации уведомлений в PWA. Он позволяет серверу отправлять клиенту поток данных в формате текста или JSON без необходимости постоянного опроса сервера со стороны клиента.

SSE особенно полезны для PWA, так как позволяют уменьшить количество запросов на сервер и тем самым улучшить производительность приложения. Когда клиент отправляет запрос на сервер, чтобы получить данные, сервер может отправить большой объем информации, который может быть использован для обновления множества различных частей приложения.

Применение SSE в PWA может быть использовано для реализации различных функциональных возможностей, таких как уведомления о новых сообщениях, обновлениях контента или прогресса выполнения задач на сервере. SSE также могут использоваться для получения данных в режиме реального времени, что может быть полезно для приложений, связанных с финансовыми рынками или торговлей.

В PWA с использованием SSE для обновления информации в режиме реального времени, приложение может использовать только одно соединение с сервером, что снижает нагрузку на сервер и увеличивает производительность приложения. Кроме того, SSE могут

работать в фоновом режиме, что позволяет приложению получать уведомления и обновления в то время, когда приложение не активно.

В целом, использование SSE для реализации уведомлений в PWA позволяет увеличить производительность приложения, сократить количество запросов на сервер, а также добавить возможность получения информации в режиме реального времени.

3. Long polling

Long polling (длительное ожидание) — это еще один метод взаимодействия между клиентом и сервером в рамках PWA, который позволяет реализовать мгновенное обновление информации без необходимости перезагрузки страницы.Суть long polling заключается в том, что клиент отправляет запрос на сервер, который ожидает, пока не появятся новые данные. При этом сервер не сразу отправляет ответ, а ждет, пока не появятся новые данные или не истечет заданный таймаут. Если данные появляются, то сервер отправляет их клиенту в ответ на запрос и сразу же открывает новое соединение для дальнейшего ожидания новых данных.

Таким образом, при использовании long polling, клиент и сервер могут обмениваться данными в реальном времени без необходимости постоянно отправлять запросы на сервер.

В рамках PWA long polling может использоваться для реализации мгновенного обновления информации на странице приложения, например, для получения новых сообщений в чате или для отображения новых данных в реальном времени. Однако, необходимо учитывать, что использование long polling может привести к большому количеству запросов на сервер, что может негативно сказаться на производительности и затрате трафика. Поэтому, перед использованием этого метода, необходимо тщательно оценить потенциальную нагрузку на сервер и трафик.

4. COMET

COMET — это набор технологий, которые позволяют серверу отправлять данные на клиентскую сторону приложения, не дожидаясь запроса от клиента. В отличие от обычного подхода, когда клиент делает запрос и получает ответ, используя COMET, сервер может отправлять данные на клиентскую сторону приложения в любой момент, что делает этот подход особенно полезным для реализации уведомлений и обновлений данных в режиме реального времени.

В контексте PWA, COMET может использоваться для реализации функциональности, которая требует непрерывной связи между сервером и клиентом, такой как уведомления и обновления контента в режиме реального времени. Например, приложение новостей может использовать COMET для получения обновлений новостей с сервера и мгновенного отображения их на странице без необходимости перезагрузки страницы. Кроме того, COMET может использоваться для создания чата или онлайн-игр, где необходимо обеспечить быстрое и надежное обмен сообщениями между сервером и клиентом.

Одним из наиболее распространенных протоколов, используемых в COMET, является протокол длинных опросов (long-polling), который является более эффективным, чем традиционный опрос, так как он позволяет сократить задержку между запросом клиента и ответом сервера. Вместо того, чтобы клиент делал множество запросов, сервер удерживает подключение открытым, возвращая ответ только тогда, когда появляются новые данные. Это

позволяет сократить количество запросов, что в свою очередь уменьшает нагрузку на сервер и снижает задержки в обмене данными.

5. Service Worker

Service Worker является одним из ключевых компонентов в создании прогрессивных веб-приложений (PWA). Он позволяет создавать мощные приложения, которые могут работать в офлайн-режиме, а также значительно ускоряет работу приложения в онлайн-режиме.

Service Worker — это скрипт, который работает в фоновом режиме, независимо от того, открыта ли страница или нет. Он может перехватывать запросы, управлять кэшированием ресурсов и даже отправлять уведомления. Благодаря этому Service Worker может значительно повысить производительность приложения и уменьшить его зависимость от сети.

Одним из наиболее полезных применений Service Worker является кэширование ресурсов приложения. После первого запуска PWA, Service Worker может кэшировать необходимые для работы ресурсы, такие как HTML, CSS, JavaScript, изображения и другие. При следующем запуске приложения, эти ресурсы будут загружаться из кэша, что уменьшит время загрузки и снизит нагрузку на сервер.

Кроме того, Service Worker может работать в фоновом режиме и обновлять кэш в фоновом режиме. Это позволяет обновлять приложение, даже если оно не запущено на устройстве пользователя. Кроме того, Service Worker может проверять наличие обновлений и загружать их, даже если пользователь не обновляет страницу.

Важно отметить, что Service Worker работает только через протокол HTTPS, что обеспечивает безопасность и защиту от атак. Кроме того, использование Service Worker может значительно улучшить опыт работы пользователей, ускорив загрузку страниц и повысив отзывчивость приложения в целом.

Таким образом, Service Worker — это мощный инструмент для создания прогрессивных веб-приложений, который может значительно улучшить производительность приложения и повысить опыт работы пользователей.

6. Web App Manifest

Web App Manifest — это JSON-файл, который описывает приложение и его свойства, такие как имя, иконки, цвет темы и многое другое. Этот файл используется для настройки PWA, чтобы приложение могло быть установлено на устройство и работать как нативное приложение.

Web App Manifest позволяет создавать иконки для приложения разных размеров, чтобы они могли отображаться на разных устройствах. Он также позволяет задавать цвета для элементов пользовательского интерфейса, настройки ориентации экрана и многое другое.

С помощью Web App Manifest можно задавать параметры, необходимые для корректного отображения приложения на мобильных устройствах, такие как режим отображения приложения на весь экран, блокирование масштабирования, настройка ориентации экрана и другие параметры.

Web App Manifest является важной частью PWA, так как он позволяет пользователям добавлять приложение на свой главный экран и использовать его без доступа к Интернету. Это повышает удобство использования приложения и улучшает опыт пользователя.

Заключение

В статье были рассмотрены основные аспекты и возможности создания PWA (Progressive Web Applications) — веб-приложений, обладающих функциональностью и возможностями мобильных приложений. Были рассмотрены такие ключевые технологии, как Service Worker и Web App Manifest, а также рассмотрены возможности использования WebSocket, Server-Sent Events, Long Polling и Comet в рамках PWA. Были представлены преимущества и недостатки использования каждой из технологий, а также даны рекомендации по выбору наиболее подходящего подхода в зависимости от конкретных задач и требований к приложению.

Литература

1. Закас Н. JavaScript. Оптимизация производительности / Н. Закас. – Пер. с англ. – СПб. : Символ-Плюс, 2012. – 256 с.
2. Современный учебник JavaScript : сайт. – 2007. – URL: https://learn.javascript.ru/ (дата обращения: 17.03.2023)
3. Что такое Long-Polling, WebSockets, SSE и Comet // MyRusakov : [сайт]. – 2010. – URL: https://myrusakov.ru/long-polling-websockets-sse-and-comet.html (дата обращения: 18.03.2023)
4. MDN Web Docs – Mozilla : сайт. – 2005. – URL: https://developer.mozilla.org/ (дата обращения: 17.03.2023)

УДК 004.4, 004.7

Web-приложение “Научные публикации” с микросервисной архитектурой

Д. В. Разинков

Воронежский государственный университет

Введение

С развитием сети Интернет все больше возможностей предоставляется пользователю. Во многом это возможно с помощью web-приложений, которые позволяют выполнять действия с помощью браузера на компьютере или другом подходящем устройстве. Такие приложения должны функционировать стабильно и быть готовыми к большим нагрузкам.

Одно из направлений в разработке web-приложений связано с информационной поддержкой учета научно-исследовательской работы преподавателя высшего учебного заведения. Создание информационной системы в виде web-приложения для работы с публикациями преподавателей и студентов позволит облегчить и централизовать данный процесс.

Данное приложение должно предоставлять следующие возможности:

1. Регистрация пользователя и вход на платформу.
2. Доступ в личный кабинет.
3. Просмотр информации о публикация.
4. Создание, редактирование и удаление данных публикаций.
5. Администрирование пользователей web-приложения.

1. Этапы разработки

Разработку web-приложений можно разделить на несколько этапов, их выполнение может проходить как одновременно, так и последовательно. Их порядок и состав зависит от проекта.

1. Проектирование архитектуры и определение необходимых технологий.
2. Разработка и создание дизайн-макетов сайта.
3. Back-end разработка:
   a. Создание микросервисов.
   b. Создание инфраструктуры web-приложения.
   c. Настройка взаимодействия между микросервисами и клиентом.
4. Front-end разработка:
   a. Создание макетов страницы.
   b. Верстка страниц и шаблонов.
5. Размещение web-приложения на сервере.

2. Анализ технологий для Back-end разработки

2.1. Spring Framework

Spring Framework (spring) – универсальный фреймворк с открытым исходным кодом для Java-платформы. Из себя представляет коллекцию меньших фреймворков, концентрирующихся на определенной задаче. Позволяет создавать монолитные и микросервисные инфраструктуры.

Spring MVC обеспечивает архитектуру паттерна Model — View — Controller при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения, обеспечивая при этом свободную связь между ними. Вся логика работы Spring MVC построена вокруг DispatcherServlet, который принимает и обрабатывает все HTTP-запросы и ответы на них.

Spring Security — это модуль Spring boot, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. Отражает разрешения, выданные пользователю в масштабе всего приложения, такие разрешения (как правило, называются «роли»), например ROLE_ANONYMOUS, ROLE_USER, ROLE_ADMIN. Также хранит информации о текущем пользователе, которую можно использовать для построения логики приложения.

Spring JPA (Java Persistence API) — это спецификация Java, описывающая систему управления сохранением java объектов в таблицы реляционных баз данных в удобном виде. Сама Java не содержит реализации JPA, однако есть существует много реализаций данной спецификации от разных компаний. Примером такой реализации может послужить Hibernate. Он реализует описания JPA, но в то же время, есть дополнительные возможности, не описанные в общей спецификации. Библиотека не только решает задачу связи классов Java с таблицами базы данных, но и также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC кода. Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос приложения на любые базы данных SQL.

2.2. Микросервисная архитектура

Микросервисная архитектура — вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей — микросервисов.

Преимущества:

1. Повышенная отказоустойчивость, по сравнению с монолитной архитектурой. При отказе одного модуля остальные продолжают работать, пользователь может продолжать пользоваться web-приложением, но не с полной функциональностью.
2. Разделение разработки. Если команда проекта состоит из нескольких разработчиков, то каждый их них может работать над своим модулем и логикой, связанной с его задачей, что дает изоляцию бизнес-процессов друг от друга.
3. Возможность горизонтального масштабирования. Если web-приложение не будет справляться с нагрузкой, то есть два варианта исправления данной проблемы: увеличить количество ресурсов сервера или увеличить количество микросервисов. Второй вариант имеет следующие преимущества: можно масштабировать необходимые модули, на которые идет максимальная нагрузка; количество одновременно работающих микросервисов быстро корректируется в зависимости от нагрузки; ресурсы сервера имеют верхний лимит, что нельзя сказать о горизонтальном масштабирование.
4. Добавление новых функций системы, не затрагивая существующие процессы. Для этого достаточно добавить новый микросервис с необходимой логикой.

Недостатки:

1. Сложность системы. Необходимо настраивать связь между всеми микросервисами.
2. Проектирование микросервисной архитектуры требует точного определения назначения модуля. Если он будет иметь сильную связность с другими, то это может привести к замедлению работы web-приложения и усложнение разработки.

3. Анализ технологий для Front-end разработки

3.1. React

ReactJS - JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов, разрабатывается и поддерживается Facebook. Этот фреймворк подходит для создания огромных веб-приложений, где данные могут меняться на регулярной основе.

Преимущества:

1. Высокий уровень гибкости и быстрая отзывчивость.
2. Виртуальная DOM (document object model), которая позволяет упорядочивать документы форматов HTML, XTML или XML в дерево.
3. Открытый исходный код, который постоянно дополняется и улучшается.
4. Малое требование к ресурсам, что обеспечивает комфортное использование на большинстве клиентских машинах.

Недостатки:

1. Мало официальной документации.
2. Предназначен в основном для создания пользовательских интерфейсов, что сужает варианты использования.

Заключение

В результате работы было разработано web-приложение, которое позволяет модернизировать работу по учету научных публикаций работников высшей школы. Данное приложение предоставляет пользователю возможность создавать, редактировать и удалять данные о публикациях, доступ в личных кабинет. При разработке была использована микросервисная архитектура, которая увеличивает стабильность работы приложения. Для работы с клиентской частью был использован фреймворк React, который обеспечивает быстроту и комфортность работы с приложением.

Литература

1. Официальная документация Spring – Режим доступа: https://docs.spring.io/spring-framework/docs/current/reference/html/.
2. Официальная документация React - Режим доступа: https://react.dev/learn.
3. Официальна документация Koltin – Режим доступа: https://kotlinlang.org/docs/home.html.

УДК 004

РЕАЛИЗАЦИЯ МОДУЛЯ МОНИТОРИНГА МАРШРУТОВ ТРАНСПОРТНЫХ СРЕДСТВ

С. В. Решетов, В. М. Мельников

Воронежский государственный университет

Введение

Современный компьютер — это не только средство связи с миром, но и инструмент организации жизни и рабочей активности. Некоторые компании в сфере транспорта и доставки до сих пор пользуются устаревшими методами отслеживания транспортных средств. Зачастую отслеживание происходит только два раза при отправлении и прибытии, это затрудняет корректирование маршрута во время движения. Проблему решают приложения, которые позволяют отслеживать транспортные средства на протяжении всего маршрута в онлайн режиме.

Цель работы — провести анализ существующих решений, спроектировать модель данных, разработать серверную и клиентскую части приложения модуля мониторинга транспортных средств.

1. Постановка задачи

Спроектировать и разработать web приложение, предоставляющее набор инструментов для мониторинга транспортных средств:

- провести анализ существующих решений;
- спроектировать модель базы данных;
- реализовать серверную и клиентскую части приложения.

Приложение должно предоставлять следующую функциональность:

- добавлять транспортное средство;
- добавлять маршрут;
- добавлять контрольные точки маршрута;
- отслеживать выход за границы маршрута по времени и координатам;
- выводить сообщения об отклонениях от заданного времени;
-

2. Анализ существующих решений

Для анализа были выбраны следующие программы: Мониторинг транспорта и грузов на базе GPS и ГЛОНАСС, GPShome, Live GPS Tracking. Приложения оценивались по следующим критериям: удобство в освоении и использовании, интерфейс, тестовый период и бесплатная версия, цена, web приложение.

Мониторинг транспорта и грузов на базе GPS и ГЛОНАСС

Удобство в освоении и использовании, краткое описание: при использовании сервиса будет страница с уникальной ссылкой, где будет список отслеживаемых транспортных средств, возможность выбрать зоны. Возможность задать расписание, маршруты, отчеты, просмотр истории и наблюдения.

Интерфейс: при нажатии на кнопку «События» открывается окно со всеми транспортными средствами. У каждого транспортного средства можно задать дату начала и окончания, надо ли уведомлять в интерфейсе по смс или email и шаблон сообщения. Во вкладке наблюдение клиент может выбрать транспортное средство. Система автоматически показывает пройденный путь и где транспортное средство находится сейчас.

Недостатки: сложный и нагромождённый интерфейс.

GPShome

Удобство в освоении и использовании, краткое описание: Интерфейс рабочей программы выглядит прогрессивно и максимально приближен по своему стилю к самым популярным приложениям навигации – Яндекс и Гугл. Карты отображаются в растровом и векторном формате, рабочая область слева выполняет остальные функции.

Интерфейс: простой и понятный интерфейс, легко разобраться с тем, как работает приложение.

Недостатки: нет возможности просмотра транспортных средств, нет информировании клиента в виде смс или email, присутствуют непонятные функции.

Live GPS Tracking

Удобство в освоении и использовании, краткое описание Любое устройство, на которое установлено приложение, превращается в трекер. Программа открывает следующие возможности:

- GPS-контроль объектов;
- обозначение геозон и настройка уведомлений о пересечении их границ;
- объединение приборов в группы;
- отправка СМС-сообщений с координатами;
- фотосъемка с отправкой в систему.

Интерфейс: Live GPS Tracking — это сервис GPS-контроль объектов.Насайтеразмещаютсякнопки «Все трекеры», «Подключить трекер», «Управление трекерами», нажав на которые клиент может просмотреть все трекеры, задать настройки трекера и добавить новый трекер.
Недостатки: Много лишней информации, сложно сразу разобраться в программе.

Результаты сравнения приложений представлены в таблице (табл. 1).

Таблица 1. Сравнение приложений по критериям

Критерии/Приложения	Мониторинг транспорта и грузов на базе GPS и ГЛОНАСС	GPShome	Live GPS Tracking
Удобство в освоениии использовании, краткое описание	+	+–	–
Интерфейс	+ –	+	+
Тестовый период и бесплатная версия	15 дней	–	–
Цена	15 руб. в сутки	70 руб. в сутки	90 руб. в сутки
Недостатки	Сложный и нагроможднный интерфейс	Нет возможности просмотра транспортных средств, нет информировании клиента в виде смс или email	Сложно разобраться в программе

3. Было решено создать приложение, которое бы имело простой и понятный интерфейс, без лишних данных, с возможностями просматривать сообщение об нахождении транспортных средств во время маршрута.**Средства реализации**

Для решения задачи выбраны следующие средства:

- язык разработки Java;
- фреймворк Spring Framework;
- СУБД PostgreSQL;

- фреймворк React.

В качестве языка программирования для серверной части был выбран язык Java. Возможности данного языка предоставляют все необходимые инструменты для реализации поставленныхв данной работе целей и задач. При реализации серверной части использовалась среда разработки Intellij Idea.

В качестве фреймворка для клиентской части был выбран React. Возможности данного фреймворка предоставляют все необходимые инструменты для реализации поставленныхв данной работе целей и задач. При реализации клиентской части использовалась среда разработки Visual Studio Code

4. Требования к аппаратному и программному обеспечению

Приложение предназначено для использования на IBMPC- совместимых компьютерах с операционной системой Windows 7, Windows 8, Windows 10.
Минимальные системные требования приложения:

- процессор с частотой 1,6 ГГЦ;
- 1 Гб оперативной памяти;
- 10 Мб дискового пространства для установки программы.

5. Модель данных

Модель данных представлена на рис. 1.

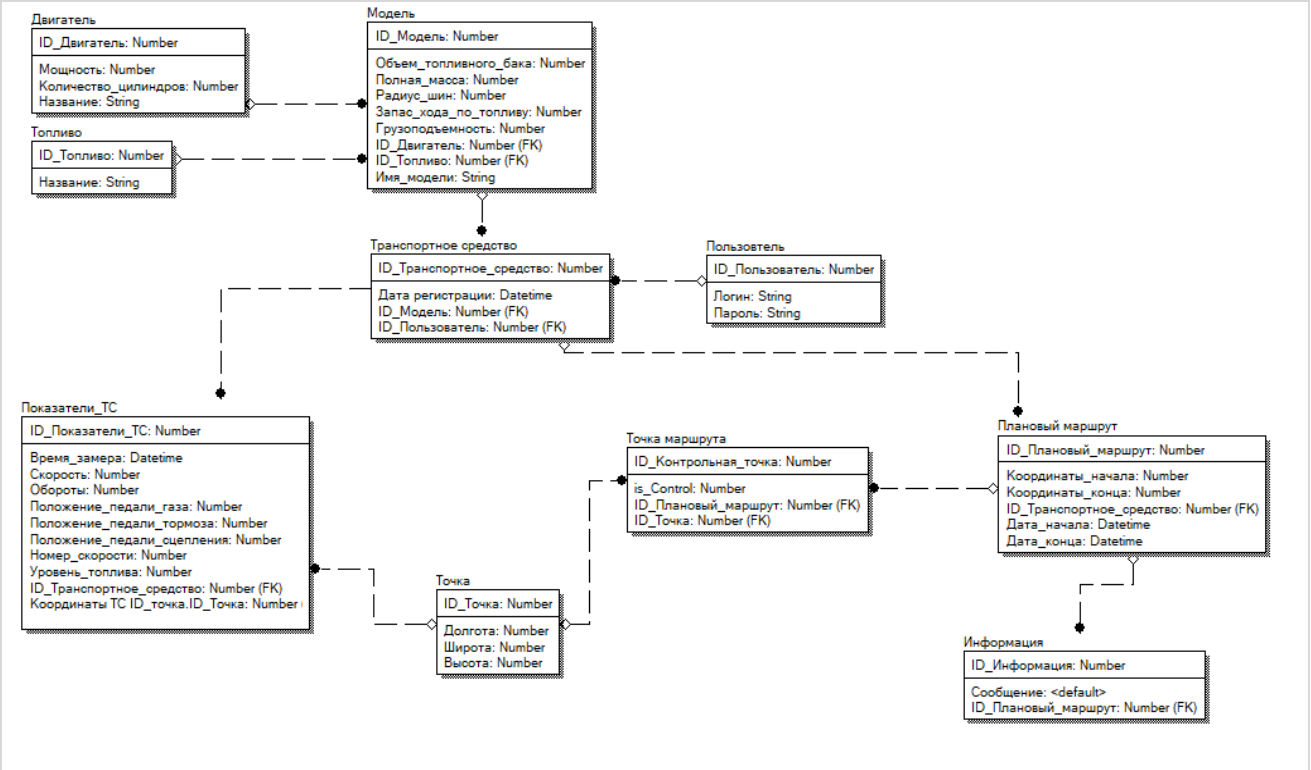


Рис. 1. Модель данных

В БД должна фиксироваться информация о пользователях, транспортных средствах, на каких маршрутах они находятся, какие показатели снимаются во время движения, и какая информация собрана по определенному транспортному средству.

6. Реализация

Методы контроллера сущности пользователя:
@PostMapping
ResponseEntity<Users> logging(@RequestBody UserDto userDto)
Проверка клиента по логину и паролю.

Методы реализации сервиса сущности пользователя:
Users checkUser(UserDto userDto)
Метод проверяет есть данный пользователь в базе данных и возвращает его, если нет, то null.

Методы контроллера сущности информация:

- *@GetMapping("{carId}")*
  ResponseEntity<Iterable<Info>> getInfo(@PathVariable Long carId)
  Получение информации по id выбранного транспортного средства;

- *@GetMapping("/parser")*
  ResponseEntity<String> getWorkParser()
  Вызов парсера для файла, в котором находятся показатели транспортного средства;

- *@GetMapping("/delete")*
  ResponseEntity<String> deleteAllInfo()
  Удаление информации из базы данных.

Методы реализации сервиса сущности информация:

- *@Override*
  public String deleteAllInfo()
  Удаление информации из базы данных;

- *private void savePlanRoute(List<Measurement> measurements)*
  Сохранение в базу данных информации о маршруте;

- *@Override*
  public String parseCSV()
  Считывание из файла по одной записи для и последующая проверка и запись в базу данных;

- *private void createInfo()*
  Создание информационного сообщения об нахождении транспортного средства;

- *private double getDistance(Point p1, Point p2)*
  Получение дистанции между двумя точками в метрах;

- *private boolean checkCarPosition(PlanRoute planRoute, Measurement measurement)*
  Проверка транспортного средства, если более 10 метров, то будет создано сообщение о схождении с маршрута;

• *@Override*
public List<Info> getAllInfo(Long carId)
Получение всех информации хранимой в базе данных для выбранного транспортного средства.
Диаграммы классов для User и Info изображена на рис. 2.1 и рис. 2.2.

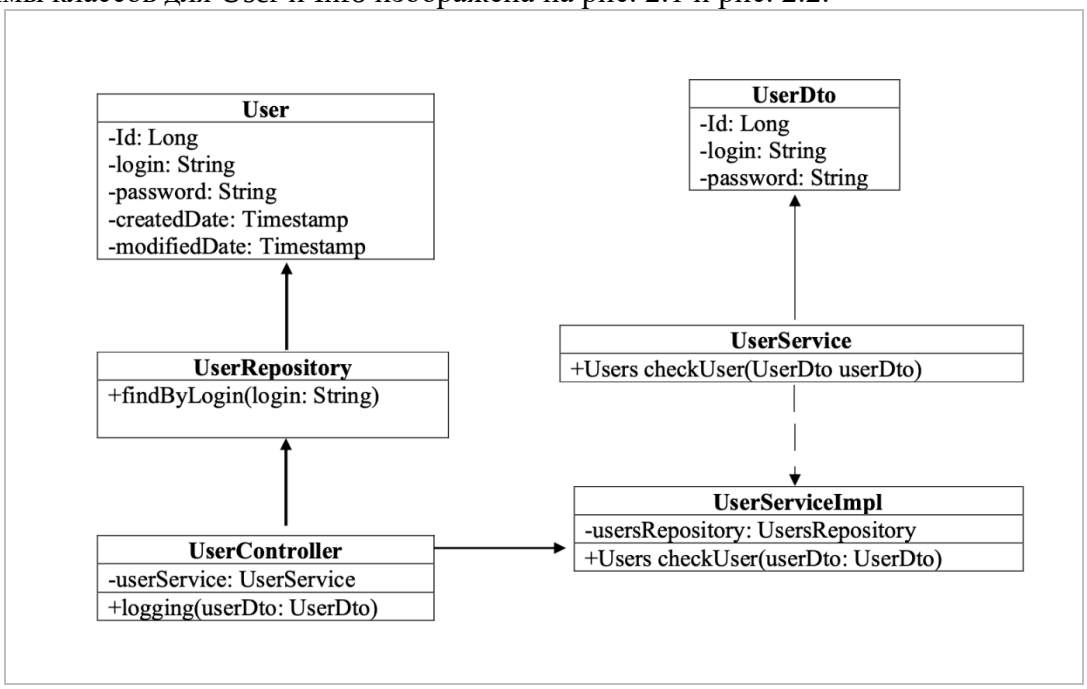


Рис. 2.1 Диаграмма класса User

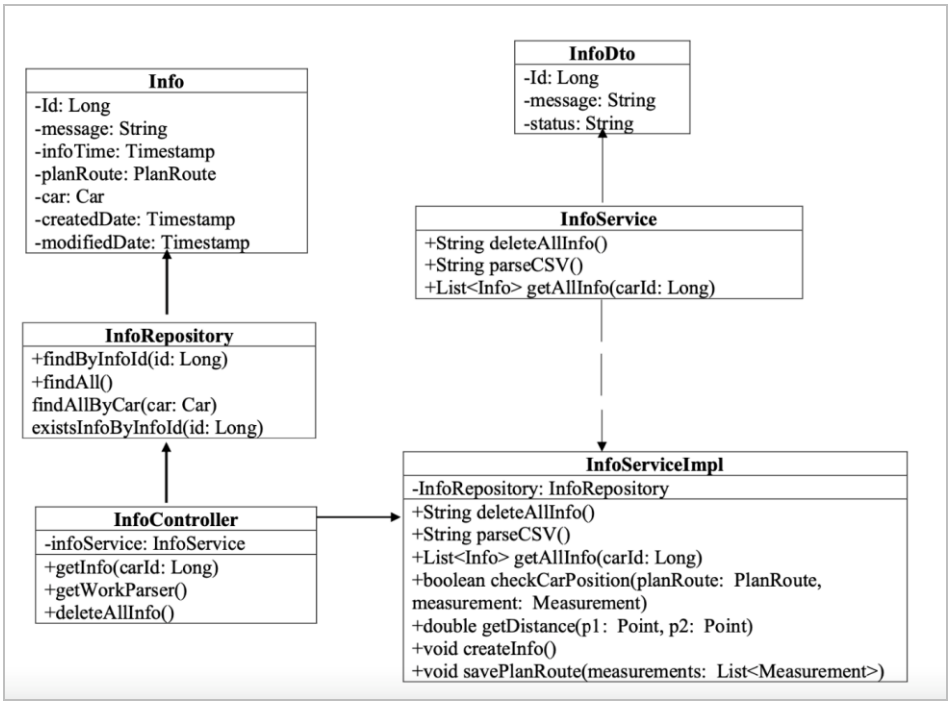


Рис. 2.1 Диаграмма класса Info

Заключение

В работе была рассмотрена структура модуля мониторинга транспортных средств, был проведен анализ существующих решений, была спроектирована модель базы данных, была

реализована серверная и клиентская частей приложения, а в дальнейшем планируется добавление новых транспортных средств, добавление собственных маршрутов.

Список литературы

1. Блох Д. Java Эффективное программирование / Д. Блох. – Москва : Лори, 2016. –440 с.
2. Васильев А. Н. Программирование на Java для начинающих / А. Н. Васильев. – Москва : Эксмо, 2014. – 416 с.
3. Герман О. В. Программирование на Java и C# для студентов / О. В. Герман. –Санкт-Петербург : BHV, 2005. – 512 с.
4. Давыдов С. IntelliJIDEA Профессиональное программирование на Java / С. Давыдов. – Санкт-Петербург : BHV, 2005. – 800 с.
5. Нимейер П. Программирование на Java / П. Нимейер, Д. Леук. – Москва : Эксмо, 2018. – 448 с.
6. Савитч У. Язык Java. Курс программирования / У. Савитч. – Москва : Вильямс, 2015. – 928 с

УДК 004.4+004.056.5

АНАЛИЗ ФРЕЙМВОРКА LARAVEL КАК ИНСТРУМЕНТА ДЛЯ РАЗРАБОТКИ БЕЗОПАСНЫХ WEB-САЙТОВ

А. Д. Рязанов, В. П. Железняк

Воронежский государственный университет

Введение

Современный мир невозможно представить без web-сайтов. При этом требования к ним растут с каждым годом. Сайты должны быть безопасными, быстрыми, удобными, функциональными и современными. Для упрощения их разработки используются различные технологии и фреймворки.

Фреймворк — это платформа, определяющая структуру программного продукта. Веб-фреймворк — это платформа, преимущественно использующая язык PHP, определяющая структуру веб-сайта.

Существует огромное количество различных веб-фреймворков, в настоящей статье рассмотрим фреймворк Laravel.

Цель научной статьи: провести анализ фреймворка Laravel.

Задачи:

1. Рассмотреть возможности Laravel в области защиты информации.
2. На основе анализа возможностей Laravel в области защиты информации определить преимущества и недостатки.
3. Реализовать проект “Computer Helpers” с использованием фреймворка Laravel и его возможностей в области защиты информации

1. Общие сведения о фреймворке Laravel

Laravel — это открытый и бесплатный веб-фреймворк с открытым исходным кодом, который используется для разработки веб-сайтов. Он был создан в 2011 году Тейлором Отвеллом и предоставляет разработчикам высокоуровневые инструменты для создания быстрых, безопасных и эффективных веб-сайтов.

2. Возможности Laravel в области защиты информации

Фреймворк Laravel имеет большое количество функций и механизмов, которые помогают разработчикам создавать безопасные web-сайты.

Рассмотрим несколько из них.

2.1.Аутентификация и авторизация

Laravel предоставляет функциональность аутентификации и авторизации, которая позволяет разработчикам создавать безопасные системы управления пользователями. Эти механизмы помогают идентифицировать пользователей, проверять их права доступа и защищают данные от несанкционированного доступа.

Для аутентификации в Laravel есть фасад Auth, который позволяет удобно работать с

аутентификацией пользователей на сайте.

Его методы позволяют:

1. проверять статус текущего пользователя;
2. регистрировать новых пользователей;
3. выполнять вход и выход из системы;
4. управлять правами доступа к различным ресурсам сайта.

К примеру, для аутентификации пользователя по его email и паролю можно использовать метод Auth::attempt():

Листинг 1

```
if (Auth::attempt(array('email' => $email, 'password' => $password))) {
//Если аутентификация прошла успешно, перенаправление на предыдущую страницу
  return Redirect::intended('dashboard');
}
```

Если разработчику необходима возможность запоминания входа пользователей, то он при наличии атрибута remember_token в таблице users в базе данных может использовать код:

Листинг 2

```
if (Auth::attempt(array('email' => $email, 'password' => $password), true)) {
  // Пользователь был запомнен...
}
```

Если метод Auth::attempt() вернул true, то пользователь успешно вошёл в систему и успешно был сгенерирован запоминающий токен.

Для определения того, аутентифицирован ли пользователь или нет, можно использовать метод Auth::check():

Листинг 3

```
if (Auth::check()) {
// Пользователь уже вошёл в систему...
}
```

При запоминании входов пользователей для определения того, был ли пользователь аутентифицирован с использованием cookie «запомнить меня» можно использовать метод viaRemember():

Листинг 4

```
if (Auth::viaRemember())
{
  //
}
```

Доступ к аутентифицированному пользователю можно осуществить следующим образом:

Листинг 5

```
$user = Auth::user();
```

Выход пользователя из системы также можно реализовать всего в 1 строку:

Листинг 6

```
Auth::logout();
```

Для ограничения доступа неаутентифицированного пользователя к страницам web-сайта

можно использовать middleware:

Листинг 7

```
//Ограничение доступа к одной странице
Route::get('profile',[App\Http\Controllers\UserController::class, 'index'])
->middleware('auth');

//Ограничение доступа к нескольким страницам
Route::middleware('auth')->group(function() {
   Route::get('/',[App\Http\Controllers\HomeController::class, 'index']);
   Route::get('profile',[App\Http\Controllers\UserController::class,'index']);
}
```

Если у разработчика есть необходимость в реализации многофакторной аутентификации и иных дополнительных функций, то Laravel может предложить ему такие пакеты как Laravel Fortify и Laravel Jetstream, которые позволяют быстро и эффективно добавить эти функции в проект.

Для авторизации в Laravel можно использовать шлюзы (Gates) и политики (Politicies).

Шлюз — это просто замыкание, которое определяет, имеет ли пользователь право выполнять указанное действие. Его задача — запрещать все действия, которые не разрешены политикой.

Политики — это классы, которые организуют логику авторизации (разрешения) для конкретной модели или ресурса.

Шлюзы и политики часто используются вместе, чтобы обеспечить безопасность сайта. Например, можно использовать шлюз, чтобы проверить, имеет ли пользователь права на доступ к странице, а затем использовать политику, чтобы проверить, имеет ли пользователь права на доступ к определенным действиям на этой странице.

Пример использования шлюза и политики:

Листинг 8

```
// создаем новый шлюз
php artisan make:middleware AccessMiddleware
// определяем логику шлюза в файле AccessMiddleware.php
public function handle($request, Closure $next)
{
   if (!$request->user()->isAdmin()) {
      abort(403);
   }
   return $next($request);
}

// регистрируем новый шлюз в файле app/Http/Kernel.php
protected $routeMiddleware = [
   ...
   'access' => \App\Http\Middleware\AccessMiddleware::class,
];

// генерируем новую политику для модели Post
php artisan make:policy PostPolicy --model=Post
// определяем права доступа в файле PostPolicy.php
public function update(User $user, Post $post)
```

```
{
   return $user->id === $post->user_id;
}

// записываем новую политику и шлюз в маршруты
Route::middleware('access')->group(function () {
   Route::get('/post/{post}', function (Post $post) {
      $this->authorize('update', $post);
      return view('post', ['post' => $post]);
   });
   Route::post('/post/{post}/update', function (Post $post) {
      $this->authorize('update', $post);
      $post->update(request()->all());
      return redirect('/post/' . $post->id);
   });
});
```

Если при разработке присутствует необходимость привязки разрешений к ролям пользователя, то в Laravel можно использовать бесплатный пакет Spatie Laravel Permission.

2.2.Хеширование паролей

Laravel предоставляет удобный способ хеширования паролей при регистрации пользователя и проверки правильности ввода при аутентификации. Для этого используется встроенный в Laravel пакет Hash.

Когда пользователь регистрируется в Laravel, его пароль хешируется с помощью метода Hash::make(), и сохраненный хеш пароля передается в базу данных. Когда пользователь входит в систему, Ларавель использует метод Hash::check(), чтобы сверить хеш введенного пароля с тем, что хранится в базе данных.

Пример использования метода Hash::make():

Листинг 9

```
$password = 'mysecretpassword';
$hashed = Hash::make($password);
```

Пример использования метода Hash::check():

Листинг 10

```
$password = 'mysecretpassword';
// Хеш, сохраненный в базе данных
$hashed='$2y$10$2Qz8wLFlVxvzVI190tu7/eRiOZKiF9Tk/xvBtDRdYtOoD01l506ey';
if (Hash::check($password, $hashed)) {
   // Пароль правильный
}
```

2.3.Шифрование данных

Шифрование является очень важным аспектом безопасности веб-сайтов. В фреймворке

Laravel инструментами для защиты данных являются механизмы шифрования и подписи.

Шифрование использует алгоритмы для преобразования данных в непонятный вид, который может быть расшифрован только с помощью ключа. В Laravel можно использовать следующие алгоритмы шифрования: AES-256, AES-128.

Пример шифрования данных в Laravel:

Листинг 11

```
// шифрование строки
$encrypted = encrypt('секретная информация');

// расшифровка данных
$decrypted = decrypt($encrypted);

// шифрование данных другим алгоритмом
$encrypted = Crypt::encryptString('секретная информация');
$decrypted = Crypt::decryptString($encrypted);

// шифрование массива данных
$encrypted = Crypt::encrypt([
   'имя' => 'Иван',
   'фамилия' => 'Иванов',
   'email' => 'ivanov@example.com'
]);
$decrypted = Crypt::decrypt($encrypted);
```

В Laravel также есть функция для подписи данных, которая позволяет гарантировать их целостность и подлинность. Это делается путем добавления цифровой подписи к данным, которые затем проверяются на основе секретного ключа, который изначально использовался для подписи данных.

Листинг 12

```
// подписывание строки
$signature = hash_hmac('sha256', 'секретная информация', 'секретный ключ');
// проверка корректности подписи
$isValid = hash_equals($signature, hash_hmac('sha256', 'секретная информация', 'секретный
ключ'));
```

Использование механизмов шифрования и подписи в Laravel позволяет защитить данные от несанкционированного доступа и повысить уровень безопасности сайта.

2.4.Защита CSRF токенами

Laravel предоставляет механизм защиты от межсайтовой подделки запросов (CSRF), который предотвращает любые запросы, поступающие с других сайтов, и гарантирует, что только доверенные источники могут отправлять запросы на сервер.

Для защиты от CSRF атак в нём используется механизм генерации и проверки CSRF токенов.

Когда пользователь отправляет POST, PUT, или DELETE запрос на сервер, включая данные формы (или тело запроса), сервер возвращает специальный CSRF токен (обычно в виде скрытого поля формы), который при следующем запросе должен быть отправлен обратно

серверу в заголовке "X-CSRF-TOKEN".

Laravel автоматически генерирует и хранит в сессии этот токен, а также проверяет валидность токена при каждом запросе. Если токен не совпадает с токеном, хранящимся в сессии, то Laravel вернет ошибку 419 (CSRF token mismatch).

Кроме того, Laravel предоставляет директиву шаблона @csrf которая генерирует HTML элемент для включения CSRF токена в форму.

Например:

Листинг 13

```
<form method="POST" action="/post">
   @csrf
   <input type="text" name="title">
   <button type="submit">Submit</button>
</form>
```

В данном примере директива шаблона @csrf генерирует HTML код для скрытого поля формы с CSRF токеном:

Листинг 14

```
<input type="hidden" name="_token" value="{{ csrf_token() }}"/>
```

Таким образом, всякий раз, когда пользователь отправляет запрос на сервер, Laravel автоматически проверяет валидность CSRF токена и обеспечивает защиту от атак.

2.5.Защита от SQL-инъекций

Laravel предоставляет функциональность для защиты от SQL-инъекций, которая позволяет предотвратить атаки на базу данных. Он предоставляет возможность использовать PDO, что помогает избежать инъекций и защищать базу данных.

Laravel предоставляет несколько механизмов для защиты от SQL-инъекций:

1. Использование подготовленных запросов (Prepared Statements) — это механизм, который позволяет отделить данные от запроса и работать с ними отдельно. В Laravel можно использовать функцию DB::select(), чтобы выполнить подготовленный запрос:

Листинг 15

```
$users = DB::select('SELECT * FROM users WHERE id = ?', [1]);
```

В этом примере значение 1 в запросе — это параметр, который будет передан отдельно в базу данных.

2. Использование ORM (Object-Relational Mapping) — это метод, который позволяет работать с базой данных, используя объектно-ориентированный подход. Laravel имеет встроенную ORM Eloquent, который позволяет создавать модели для каждой таблицы в базе данных:

Листинг 16

```
class User extends Model
{
        // ...
}
```

Затем можно использовать методы этой модели, чтобы выполнить запросы к базе данных:

Листинг 17

```
$user = User::where('name', '=', 'John')->first();
```

Этот запрос выберет первого пользователя с именем "John". Eloquent автоматически защитит запрос от SQL-инъекций.

3. Использование Query Builder — это метод, который позволяет создавать запросы к базе данных, используя цепочку методов:

Листинг 18

```
$users = DB::table('users')
    ->where('name', '=', 'John')
    ->get();
```

Этот запрос выберет всех пользователей с именем "John".

Query Builder автоматически защитит запрос от SQL-инъекций, используя подготовленные запросы.

Но самым эффективным способом защиты от SQL-инъекций является правильная валидация пользовательского ввода перед сохранением в базу данных, а также предотвращение использования небезопасных функций, таких как eval() или unserialize().

2.6.Защита от XSS-атак

Laravel также предоставляет механизм защиты от атак на межсайтовые скрипты (XSS), который предотвращает вставку вредоносных скриптов на страницы и гарантирует безопасность пользователей.

Он включает ряд функциональных возможностей для защиты от XSS-атак:

1. Шаблонизатор Blade автоматически экранирует переменные, переданные в представление на вывод с помощью выражения вывода {{}}, что предотвращает вставку вредоносного кода в HTML-код страницы.
2. Функции-помощники Laravel, такие как e() и htmlspecialchars(), также экранируют символы HTML, что предотвращает возможность вставки XSS-кода в вывод.
3. Методы валидации Laravel также выполняют экранирование введенных пользователем данных перед их использованием.

Несмотря на это, разработчики сайтов на фреймворке Laravel также должны прилагать свои усилия для защиты от XSS-атак.

Это может включать в себя:

1. Проверку введенных пользователем данных на наличие кода HTML перед экранированием их при помощи функций-помощников Laravel.
2. Использование фильтров данных для удаления вредоносных символов и предотвращения XSS-атак.
3. Регулярное обновление системы фреймворка и его пакетов, чтобы устранять уязвимости безопасности, которые могут использоваться злоумышленниками для проведения XSS-атак.

3. Преимущества и недостатки фреймворка Laravel

3.1.Преимущества

1. Включает в себя механизмы безопасности, такие как шифрование данных, хеширование паролей и защиту от SQL-инъекций, XSS и CSRF атак.

2. Предоставляет встроенный механизм аутентификации и авторизации, который позволяет быстро и легко настроить систему аутентификации и авторизации и защитить сайт от несанкционированного доступа.

3. Имеет большое сообщество активных пользователей и разработчиков, которые работают над улучшением фреймворка, созданием дополнительных пакетов, устранением ошибок и уязвимостей.

3.2.Недостатки

1. Несмотря на его широкие возможности в области защиты информации, разработчику необходимо предпринимать дополнительные усилия для защиты от взлома и атак.

2. Не предусматривает автоматического обновления системы и пакетов, что сохраняет возможные уязвимости безопасности и требует от разработчика осуществления данной процедуры вручную.

4. Реализация проекта “Computer Helpers”

Описанные выше преимущества и недостатки, возможности фреймворка в области защиты информации, такие как аутентификация и авторизация, хеширование паролей, защита от SQL-инъекций, CSRF и XSS-атак, были учтены при реализации проекта “Computer Helpers”. Его суть заключается в обеспечении возможности потенциальным клиентам сервисного центра ознакомиться с оказываемыми услугами, их стоимостью и сроком выполнения, используя сеть Интернет, и совершать интернет-заказы услуг по ремонту компьютерной техники и периферийного оборудования.

Заключение

Результатом исследования стал анализ возможностей фреймворка Laravel в области защиты информации.

В ходе исследования были выявлены преимущества и недостатки фреймворка в этой области. На основе анализа возможностей, преимуществ и недостатков, были определены необходимые средства для реализации безопасного web-сайта.

Литература

1. Аутентификация. – Режим доступа: https://laravel.ru/docs/v5/authentication?ysclid=lgjnrws0p2288270652. – (Дата обращения: 16.04.2023).
2. Авторизация. – Режим доступа: https://laravel.su/docs/8.x/authorization?ysclid=lgmeh9eyji61458717#creating-policies. – (Дата обращения: 16.04.2023).
3. Предотвращение атак CSRF. – Режим доступа: https://laravel.su/docs/8.x/csrf?ysclid=lgimnbl531888812493. – (Дата обращения: 16.04.2023).
4. Безопасность. – Режим доступа: https://laravel.ru/docs/v4/security?ysclid=lgimn5xwnj325978942. – (Дата обращения: 16.04.2023).
5. Шаблонизатор Blade. – Режим доступа: https://laravel.su/docs/8.x/blade?ysclid=lgjvztkkfg656619178. – (Дата обращения: 16.04.2023).

УДК 004.432.2

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ НА ПЛАТФОРМЕ ASP.NET CORE MVC. МОДУЛИ ДОБАВЛЕНИЯ, УДАЛЕНИЯ И РЕДАКТИРОВАНИЯ УСЛУГ

В. В. Савченко

Воронежский государственный университет

Введение

Web-приложение — это прикладное программное обеспечение, логика которого распределена между сервером и клиентом, а обмен информацией происходит по сети. Клиентская часть реализует пользовательский интерфейс, а серверная — получает и обрабатывает запросы от клиента, выполняет вычисления, формирует веб-страницу и отправляет её клиенту согласно протоколу HTTP.

Актуальность исследований в области вопросов построения web-приложений обусловлена тем, что данный вид программного обеспечения:

- перспективен, как инструмент электронной коммерции;
- предоставляет широкие возможности социального взаимодействия;

Основная цель — разработать приложение, предоставляющее сотрудникам удобный интерфейс управления услугами, а пользователям визуально понятное пользование данным функционалом.

Исходя из цели задача состоит в разработке модулей добавления, удаления и редактирования услуг со стороны клиентской и серверной части приложения.

1. Особенности платформы ASP.NET Core MVC

ASP.NET Core MVC представляет собой платформу для создания сайтов и веб-приложений с использованием паттерна (или шаблона) MVC (model-view-controller). Шаблон MVC подразумевает взаимодействие трех компонентов: контроллера (controller), модели (model) и представления (view). Рассмотрим подробнее каждый из компонентов:

- Контроллер (controller) представляет класс, с которого и начинается работа приложения. Этот класс обеспечивает связь между моделью и представлением. Получая вводимые пользователем данные, контроллер исходя из внутренней логики при необходимости обращается к модели и генерирует соответствующее представление;
- Представление (view) — это визуальная часть или пользовательский интерфейс приложения — например, html-страница, через которую пользователь, зашедший на сайт, взаимодействует с веб-приложением;
- Модель (model) представляет набор классов, описывающих логику используемых данных;

На рис. 1 приведена общая схема взаимодействия данных компонентов.

Рис. 1. Общая схема взаимодействия

Особенности ASP.NET Core MVC:

- Разделение ответственности. В MVC приложение состоит из трех частей: контроллера, представления и модели, каждая из которых выполняет свои специфичные функции. В итоге приложение будет легче поддерживать модифицировать в будущем;
- В силу разделения ответственности приложения MVC обладают лучшей тестируемостью. И мы можем тестировать отдельные компоненты независимо друг от друга;
- Соответствие протоколу HTTP. Приложения MVC не поддерживают объекты состояния (ViewState). Ясность и простота платформы позволяют добиться большего контроля над работой приложения;
- Гибкость. Вы можете настраивать различные компоненты платформы по своему усмотрению. Изменять какие-либо части конвейера работы MVC или адаптировать его к своим нуждам и потребностям;

2. Модули добавления, удаления и редактирования услуг

В приложении в качестве настроек в файле с расширением .json указывается строка подключения к локальной БД MS SQL Server.

Для реализации модулей добавления, редактирования и удаления услуг необходимо создать базовый абстрактный класс, в котором определяем ряд свойств для услуг:

- первичный ключ;
- название;
- краткое описание;
- полное описание;
- титульная картинка;
- значения метатегов (заголовок, описание, ключевые слова);
- дата создания сущности;

После этого создаем доменный объект — класс услуг на сайте, который наследуется от базового абстрактного класса с необходимыми свойствами.

Далее необходимо связать доменный объект с БД, то есть создать контекст БД, в котором будет связь логики работы web-приложения с БД (контекст БД служит для связи объектов сайта с БД).

Затем определяем интерфейс для доменного объекта, который реализует необходимое

поведение:

- выбрать все услуги;
- провалиться в какую-нибудь услугу, то есть выбрать ее по идентификатору;
- обновить или создать новую услугу;
- удалить услугу;

В результате связывает конфигурацию из файла с расширением .json с проектом, подключаем необходимый функционал приложения в качестве сервисов, подключаем контекст БД, настраиваем identity систему, в которой определяем требования системы безопасности, настраиваем authentication cookie, добавляем сервисы для контроллеров и представлений.

Теперь чтобы окончательно связать доменную модель и БД необходимо создать миграцию (это такой инструмент, который позволяет отслеживать изменения в БД в соответствиями с изменениями в кодовой базе) и обновить по этой миграции БД. После чего в MS SQL Server можем наблюдать созданную БД с необходимыми таблицами.

Для взаимодействия с услугами в панели администратора создается контроллер, в котором с помощью идентификатора в БД ищется услуга. Если услуга не найдена, то создается новая услуга, то есть объект класса, если же услуга присутствует в БД, то выбираем ее, то есть для создания и редактирования услуги служит одно представления. В представлении указывается html-форма, которая отправляется методом POST на сервер для визуализации и взаимодействия с ней.

Для удаления услуги она ищется по идентификатору и удаляется из БД.

В результате разработки получаем интерфейс для управления услугами, а именно: выбор услуг, добавление услуг, редактирование услуг, а также удаление услуг.

3. Интерфейс пользователя

На рис. 2 представлена панель администратора, на которой пользователь может добавить услугу.

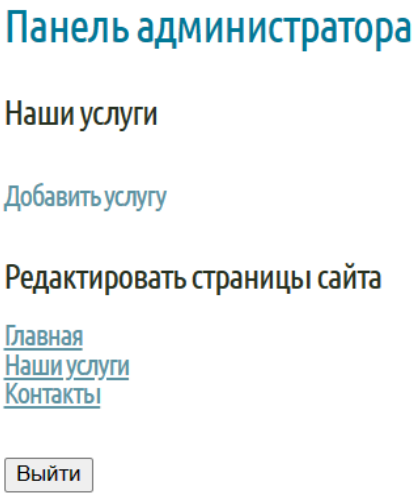


Рис. 2. Функция «Добавить услугу» на панели администратора

.

На рис. 3 после выбора функции «Добавить услугу» открывается страница добавления контента услуги, на которой пользователь описывает следующие поля:

- название услуги;
- краткое описание услуги;
- полное описание услуги;
- титульная картинка;
- метатеги;

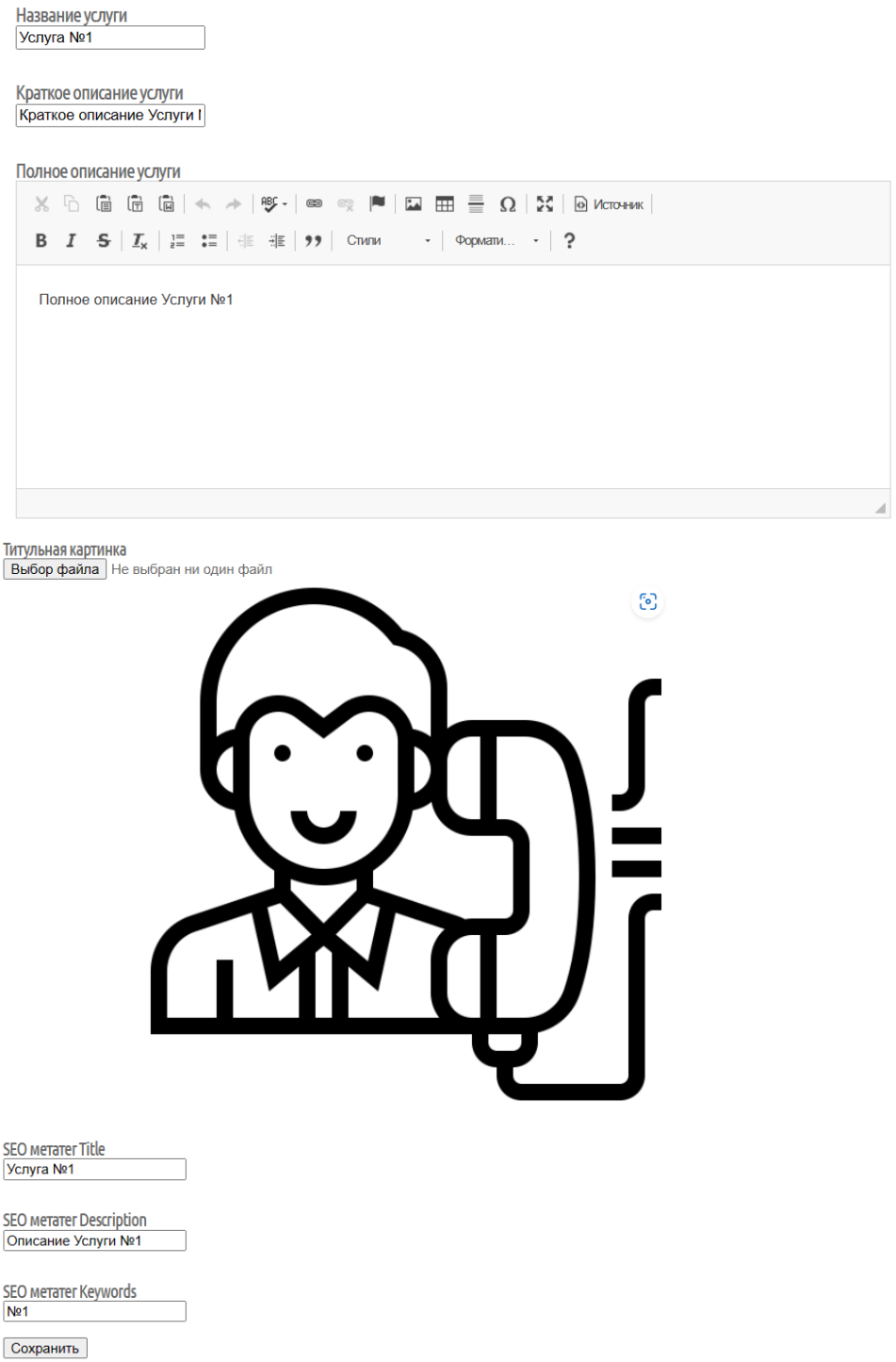


Рис. 3. Добавление контента услуги

После нажатия кнопки «Сохранить» услуга автоматически добавляется в базу данных, открывается панель администратора, на которой можно наблюдать добавленную услугу. Данная визуализация представлена на рис. 3.

Рис. 3. Панель администратора после добавления услуги.

Далее переходим к модулю редактирования услуг. Так как функционал добавления и редактирования одинаковый, то после нажатия кнопки «редактировать» открывается страница описания контента, как при добавлении услуги, на которой можно вносить изменения и правки. После нажатия кнопки «Сохранить» все изменения будут зафиксированы в базе данных и в панели администратора.

Также на панели администратора можно удалить услугу, для этого необходимо нажать кнопку «удалить», услуга исчезнет со страницы как визуально, так и запись о ней в базе данных будет удалена.

Заключение

В заключении можно отметить преимущества платформы ASP.NET Core MVC:

- Основывается на масштабируемой и продуманной модели для создания крупных веб-приложений.
- Четкое разделение задач для максимальной гибкости.
- Разделение обязанностей на основе шаблона "модель — представление — контроллер" гарантирует, что бизнес-модель можно легко развивать, не затрагивая при этом реализацию возможностей более низкого уровня.

К недостаткам можно отнести:

- У начинающих разработчиков могут быть сложности с освоением декларативного подхода.
- Небольшое количество библиотек и компонентов.
- Для запуска проекта требуется больше времени.

Таким образом данная платформа сложнее в изучении, однако позволяет сократить количество кода, связанного с разработкой интерфейса, что в целом сокращает время на создание web-приложения.

Литература

1. Агуров П. В. ASP.NET. Сборник рецептов / П. В. Агуров. – Санкт-Петербург : БХВ-Петербург, 2010. - 416 с.

2. Столбовский Д. Н. Разработка Web-приложений ASP.NET с использованием Visual Studio .NET : учебное пособие / Столбовский Д.Н.. – Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. – 375 с.

3. Умрихин Е. Д. Разработка Web приложений с использованием ASP.NET Core MVC / Е. Д. Умрихин. – Санкт-Петербург : БХВ-Петербург, 2023. - 528 с.

4. Разработка приложений MVC ASP.NET Core. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/architecture/modern-web-apps-azure/develop-asp-net-core-mvc-apps. – (Дата обращения: 20.02.2023).

5. Руководство по ASP.NET Core MVC. – Режим доступа: https://metanit.com/sharp/aspnetmvc/. – (Дата обращения: 21.02.2023).

УДК 004

АВТОМАТИЗАЦИЯ СОЗДАНИЯ ОТЧЕТОВ В САЛОНЕ КРАСОТЫ

А.В. Сакович

Воронежский государственный университет

Введение

YCLIENTS – это CRM, используемая в салоне красоты Sointera для реализации онлайн-записи и автоматизации определённых сфер деятельности.

Ежедневно 260 тыс. человек используют платформу YCLIENTS для ведения своего бизнеса. Благодаря этой программе ежемесячно создаётся более 14 млн. записей. Одним из преимуществ данной платформы является возможность обращения в техническую поддержку в режиме 24/7.

YCLIENTS обладает следующей функциональностью:

- расчет зарплат;
- финансовый учет;
- перемещения, приход, продажа, списание товара;
- запись клиентов на услуги;
- телефония и многое другое.

В данной CRM существует внутренняя аналитика, но ее представление не удовлетворяет запросам заказчика. Необходимая для единовременного анализа информация размещена на разных страницах сайта, а ее представление в Excel-документах нечитабельно и плохо подходит для анализа.

Из-за ограниченных возможностей в YCLIENTS, возникла задача получения отчетов автоматическим путем.

1. Аналитика CRM для сферы услуг

Основной целью сравнительного анализа для заказчика является выявление функциональности и производительности предприятия. Требуется приложение, которое позволяет представить отчет в наиболее понятном виде, обеспечивает возможность мобильного просмотра отчетов.

Для упрощения получения данных из программы заказчик рассматривал три возможных решения задачи:

1. Переход на иной CRM с иной аналитикой.
2. Использование сервиса ROISTST, предлагаемый компанией YCLIENTS, как партнер.
3. Создание собственного приложения.

2. Варианты решения задачи создания отчетов

На данный момент существует большое количество CRM-систем, например, 1С:Салон красоты – это многофункциональная система управления. Переход на 1С:Салон красоты мог бы разрешить проблемы в создании отчетов удобных потребителю, но является неэффективным из-за потребности в переносе большого количества информации, хранящейся в

YCLIENTS. Отсутствие автоматического переноса данных на другую платформу делает переход на новую CRM невозможным.

Рынок внешних программ для аналитики также достаточно разнообразен, например, Roistat, система сквозной бизнес-аналитики, партнер YCLIENTS. Она выполняет комплексную оценку деятельности предприятия, выявляет основные тенденции его развития, рассчитывает базовые нормативы для планирования и прогнозирования. За секунды строит отчеты, на которые у людей уходит по несколько часов. Позволяет понять, окупаются ли ваши вложения в рекламу, и отследить источники прихода клиентов.

В Roistat большой спектр показателей, но сложность работы, большой уклон в экономическую часть и неудобное представление информации не устраивает заказчика.

Таким образом, задача разработки собственного приложения является актуальной. Приложение сможет работать с данными, предоставляемыми, действующей CRM, а также удовлетворить потребности заказчика в форме и содержании отчета.

3. Приложение для генерации отчетов

На текущий момент создание отчетов является достаточно трудоёмким процессом.

Для получения исчерпывающей и полной информации на основании данных из раздела «Аналитика» программы YCLIENTS приходится вручную создавать Excel файлы, для того чтобы выполнить действия:

- получить сведение различной информации с отдельных разделов в один документ;
- отследить динамику развития бизнеса за определенные период;
- отследить потоков клиентов.

Вся аналитика доступна для скачивания в формате Excel файла, но информация имеет нечитабельный вид, а в отдельных случаях объединяет данные, что искажает реальную статистику (на рис. 1 приведена часть выгрузки данных о записях клиентов за месяц май 2022).

Время визита	Создал		Статус визита	Источник
	Имя	Дата		
2022-05-31 19:30:00	Клиент	2022-05-31 14:34:43	Клиент пришёл	"Форма компании" новый виджет
2022-05-31 18:00:00	Клиент	2022-05-23 22:36:20	Клиент пришёл	"Форма компании" новый виджет
2022-05-31 18:00:00	Анастасия	2022-05-28 20:15:17	Клиент пришёл	Web-интерфейс для администратора
2022-05-31 18:00:00	Дарья	2022-05-27 13:16:50	Клиент пришёл	Web-интерфейс для администратора

Рис. 1 Отчет в Excel YCLIENTS

Поэтому было принято решение о составление отчетов вручную, что включает в себя сбор информации из программы и статистики, обработку и внесение данных в формат, продемонстрированный на рис. 2, заранее согласованный с заказчиком.

Показатель	Май																
	День																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Новые клиенты:	4	1	0	5	5	3	0	3		3	1	1	3	5	1		4
-инсаграмм	1	1												1			
-Вконтаке																	
-интернет	1																
-за мастером																	
-вывеска														1			
-знакомые	1									1							1
-прочее	1			5	5	3		3		2	1	1	3	3	1		3
Всего записей(пришли)	19	21	13	25	23	37	31	22		33	12	27	31	22	21		28
Запись онлайн	11	10	12	11	11	17	11	10		15	4	15	10	13	9		11
Запись Инстограмм/ВК	1			5	1		1	1		1		1	1				1
Запись ресепшн	7	11	1	9	11	20	19	11		17	8	11	20	9	12		16
Не пришел			1		1	1		1						1			2
Кол. обслуж. Клиентов	19	21	13	25	23	36	30	20		33	12	25	31	22	21		28
из них визаж. брови				1	2	2	6	3		5		6	4				
из них парикмах. зал	14	13	11	21	14	25	18	12		18	8	16	19	15	14		19
из них ногтевой зал	5	8	2	3	7	9	6	5		10	4	3	8	7	7		9

Рис. 2 Пример отчета «Учет клиентов»

Работа над созданием таких отчетов, занимает большое количество времени, что является неудобным для заказчика, так как замедляет бизнес процесс.

Для разработки приложения найдено несколько вариантов получения информации: брать данные на прямую с сайта или использовать Excel файлы, скаченные с YCLIENTS.

При детальном анализе, принято решение использовать обращение к внутреннему API площадки.

Для получения данных с интернет площадки YCLIENTS используются внутренние API. Программа клиент отправляет специально сформированный запрос на сервер и получает ответ в виде JSON. Для отправки и обработки запросов используются библиотеки языка C#. После получения запроса, данные десериализуются в объектное представление. Например:

```
public record ReportModel(
[property: JsonPropertyName("content")] string Content,
[property: JsonPropertyName("paging")] string Paging,
[property: JsonPropertyName("count")] int Count,
[property: JsonPropertyName("success")] bool Success,
[property: JsonPropertyName("error")] string Error,
[property: JsonPropertyName("extra_data")] string ExtraData
);
```

Поле Content содержит данные для отображения на странице. Остальные поля носят служебную информацию и программой не обрабатываются.

Заключение

Приложение для генерации отчетов является важным инструментом для салона красоты Sointera. Оно позволяет автоматизировать процесс создания отчетов, что значительно экономит время, деньги и упрощает анализ данных. Также создание такого приложения позволяет оптимизировать бизнес-процессы и улучшить качество обслуживания клиентов.

Литература

1. Петцольд, Ч. Программирование для Microsoft Windows на C# – В 2-х томах - Том 1. / Ч. Петцольд - пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2002. – 576 с.
2. Microsoft Excel 2013. Библия пользователя / пер. с англ. и под ред. Н. В. Воронина. - Москва [и др.] : Диалектика, 2015. – 928 с

УДК 004.75

О СПОСОБАХ РЕШЕНИЯ ПРОБЛЕМЫ НЕДОСТУПНОСТИ ОРКЕСТРАТОРА САГИ В РАСПРЕДЕЛЕННОЙ СИСТЕМЕ

А. И. Светашов

Воронежский государственный университет

Введение

В распределенных системах существуют такие системы, для которых критически важно поддержание согласованности данных и доступности системы. Например, в банковской сфере, электронной коммерции или в системах бронирования заказов. Одним из популярных подходов для обеспечения этих свойств является использование шаблона Сага (Saga), который широко используется в архитектуре микросервисов (слабо связанных программных модулей, расположенных на отдельных узлах или выполняющихся в отдельных виртуальных машинах). Однако Saga предполагает наличие узла-оркестратора в системе, и при недоступности только одного оркестратора, система целиком не будет способна выполнять запросы пользователей.

Эта статья посвящена способам решения проблемы недоступности оркестратора Saga. Будет рассмотрено использование резервных оркестраторов Saga и способы обеспечения высокой доступности оркестратора.

1. Описание шаблона

1.1. Проблема согласованности данных

В распределенных системах распространен подход, в котором каждый сервис имеет отдельную базу данных. Это позволяет иметь слабо связанные сервисы, и каждый из них может использовать разных поставщиков СУБД. При таком подходе возникает проблема согласованности данных между сервисами. В каждой отдельной службе данные будут в согласованном состоянии при использовании транзакций. Но между сервисами данные не будут согласованы.

Эта проблема может быть решена использованием протокола двухфазной фиксации (2PC) или использованием шаблона Saga. Протокол двухфазной фиксации предполагает синхронные операции, в связи с чем может произойти отказ для запроса, который затрагивает множество сервисов. Шаблон Saga использует асинхронные операции, что позволяет избежать такой проблемы.

1.2. Архитектура Саги

Шаблон Saga — это шаблон проектирования, используемый в распределенных системах для управления транзакциями, включающими несколько микросервисов. В распределенной системе транзакция может включать в себя несколько субтранзакций в разных микросервисах, и если какой-либо из этих сервисов дает сбой или не может выполнить свою часть транзакции, система может остаться в несогласованном состоянии. Saga позволяет справиться с этой проблемой, разбив сложную транзакцию на ряд более мелких независимых шагов или субтранзакций, каждая из которых управляется отдельным сервисом. Каждая вложенная

транзакция выполняется независимо, и в случае сбоя какой-либо из них, шаблон Saga предоставляет способ отката завершенных вложенных транзакций для обеспечения согласованности в системе.

В шаблоне сагой называется долгоживущая транзакция, охватывающая несколько микросервисов. Каждый микросервис, участвующий в саге, выполняет локальную транзакцию, которая обновляет собственные данные и взаимодействует с другими микросервисами для координации общей транзакции. В случае сбоя какой-либо из субтранзакций сага выполняет компенсационное действие, чтобы откатить завершенные субтранзакции и восстановить систему в согласованное состояние.

Шаблон Saga можно использовать в любой распределенной системе, где важна согласованность данных, включая финансовые системы, приложения электронной коммерции и системы онлайн-бронирования. Используя шаблон Saga, возможно обеспечение надежного и согласованного выполнения сложных транзакций даже в случае сбоев или ошибок.

Существует два основных подхода к реализации шаблона Saga.

1. Подход, основанный на оркестрации.
2. Подход на основе хореографии.

В случае оркестрации каждый микросервис, участвующий в транзакции, взаимодействует напрямую с другими микросервисами, чтобы определить правильную последовательность субтранзакций. Каждый микросервис инициирует собственную локальную транзакцию. Сервисы коммуницируют между собой для выполнения всей саги, при этом каждый микросервис знает свои обязанности и зависимости от других микросервисов.

Подход, основанный на хореографии, часто используется для относительно простых транзакций, которые содержат всего несколько субтранзакций и включают ограниченное количество микросервисов. Этот подход часто легче реализовать, чем подход, основанный на оркестровке, поскольку он требует меньшей координации между микросервисами. Пример схемы взаимодействия компонентов системы, основанной на хореографии, представлен на рис. 1.

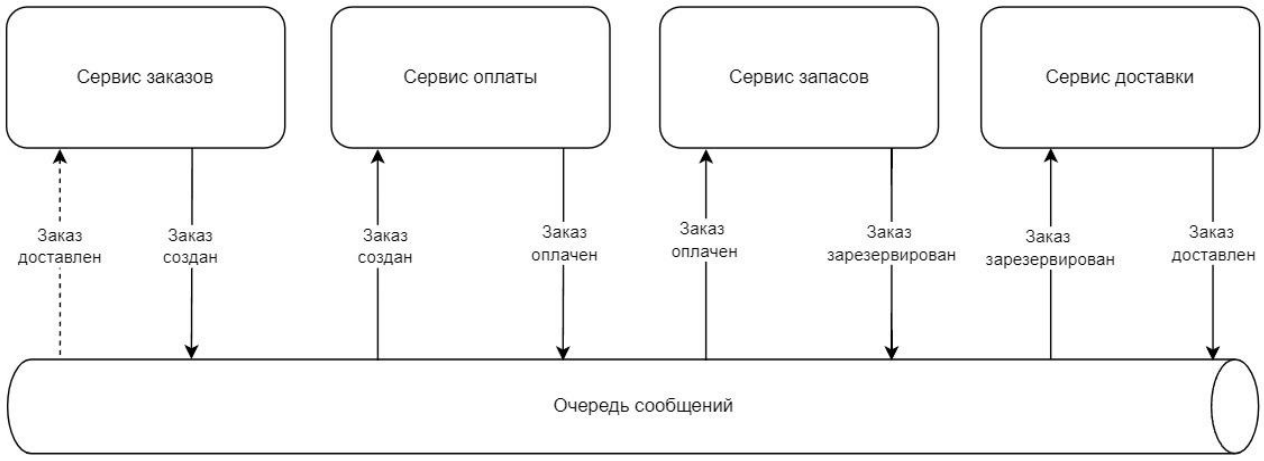


Рис. 1. Схема взаимодействия компонентов системы, основанной на хореографии

В данном подходе в случае отказа одного из сервисов, остальные сервисы должны самостоятельно определить порядок и время выполнения компенсационных действий.

В подходе, основанном на оркестрации, центральный оркестратор отвечает за управление сагой и координацию субтранзакций, выполняемых каждым микросервисом. Оркестратор связывается с каждым сервисом для инициирования и мониторинга субтранзакций. Если какая-либо из субтранзакций завершается сбоем, он выполняет необходимые компенсационные действия для отката завершенных субтранзакций

и приведения системы в согласованное состояние.

Подход, основанный на оркестрации, часто используется для более сложных транзакций, включающих несколько субтранзакций и микросервисов. Этот подход может быть более гибким, поскольку оркестратор может быть разработан для обработки различных типов транзакций и для динамического управления последовательностью субтранзакций. Схема взаимодействия сервисов для оркестрации представлена на рис. 2.

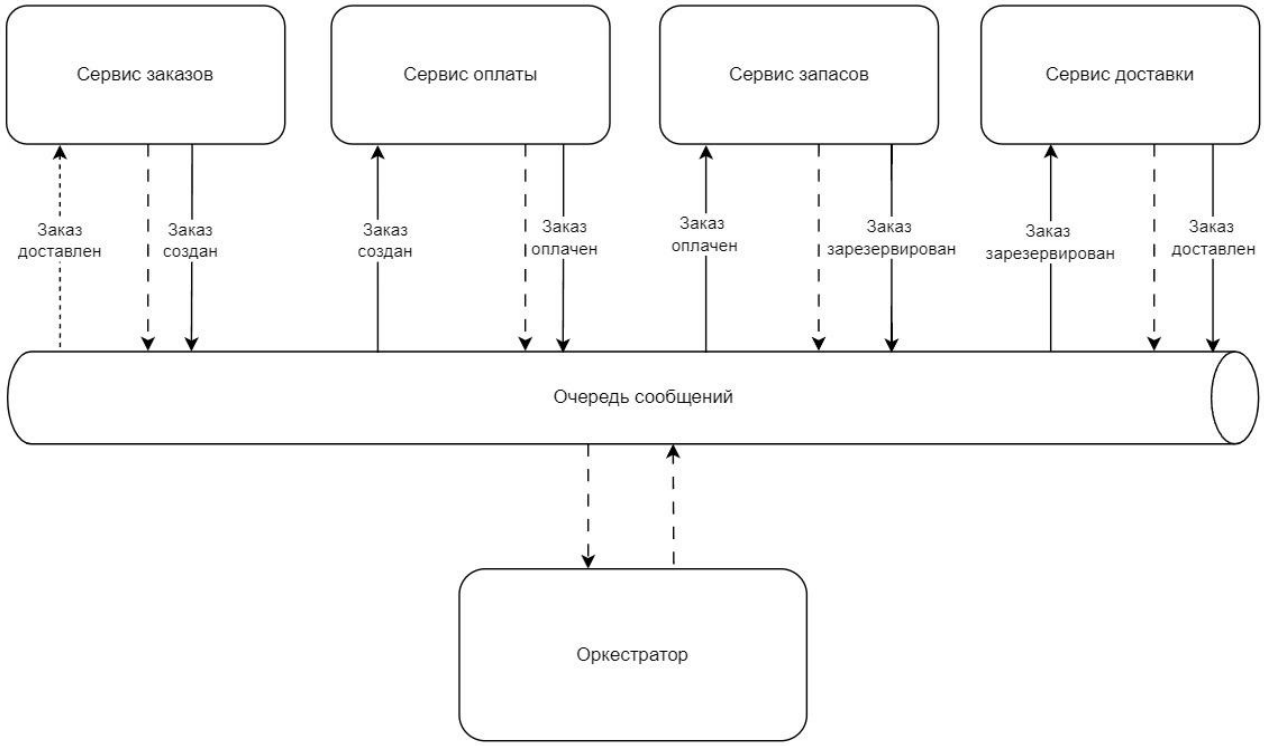


Рис. 2. Схема взаимодействия компонентов системы, основанной на оркестрации

Пунктирными линиями без подписей на схеме обозначены сообщения для оркестратора о регистрации компенсационных действий данного сервиса. В случае сбоя следующего микросервиса, оркестратор отправит сообщения всем зарегистрированным сервисам о применении компенсационных действий.

В дальнейшем будем рассматривать схему саги с использованием оркестратора для отделения логики управления сагой.

1.3. Недоступность сервисов системы

При использовании саги с оркестратором недоступность одного из сервисов не приведет к отказу в обслуживании запросов пользователей. Оставшиеся доступные сервисы способны самостоятельно выполнять запросы пользователей, не связанные с недоступным сервисом. Однако, если пользователь выполняет запрос, затрагивающий один из недоступных сервисов, оркестратор саги сможет обеспечить откат уже выполненных действий в работающих сервисах.

Например, если на приведенной схеме (рис. 2) сервис запасов перестанет отвечать на запросы, а сервис заказов создаст заказ, и сервис оплаты зарезервирует деньги покупателя, оркестратор саги по истечении заданного времени инициирует выполнение компенсационных действий для сервисов заказов и оплаты. Деньги вернутся на счет покупателя, а заказ будет

отменен. После этого система будет в согласованном состоянии. Предполагается, что очередь сообщений является отказоустойчивой.

При недоступности оркестратора возникает проблема, в случае если сервис заказов уже создал заказ, а в сервисе оплаты произошел сбой. Она заключается в том, что дальнейшие действия по откату изменений в сервисе заказов не будут выполнены, так как сервис заказов не получит сообщения от оркестратора о необходимости выполнения компенсационных действий.

Рассмотрим способы решения проблемы недоступности оркестратора саги.

2. Репликация оркестратора

Для поддержания возможности выполнять сагу, создадим несколько экземпляров оркестратора. В таком случае, если откажет один оркестратор, управлять транзакциями будет другой экземпляр оркестратора. Данная возможность реализована во многих программных комплексах для автоматизации, развертывания и масштабирования приложений. Одним из самых популярных на данный момент является Kubernetes. С его помощью можно обеспечить постоянную работу нескольких экземпляров оркестратора. Схема взаимодействия компонентов в системе приведена на рис. 3.

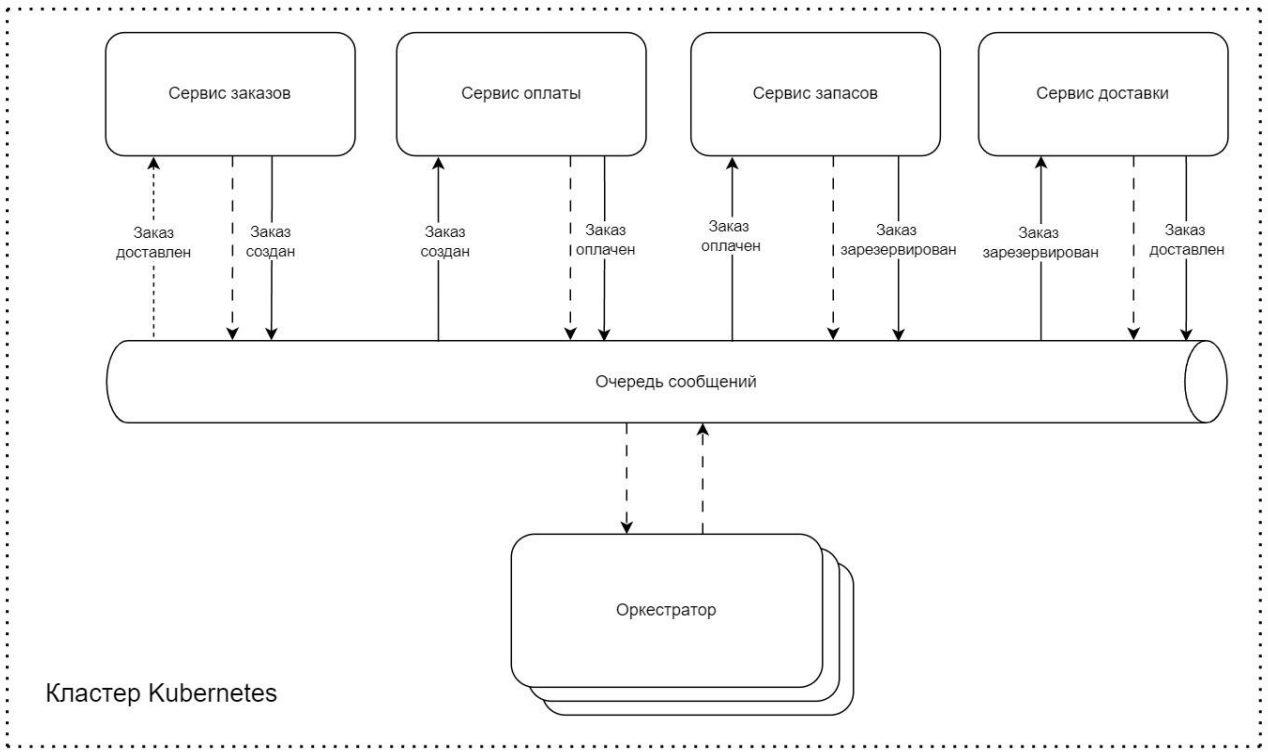


Рис. 3. Схема репликации оркестраторов в кластере Kubernetes

При такой реализации возникнет проблема согласованности состояний самих оркестраторов. Каждый оркестратор должен хранить информацию о выполняемых сагах в оперативной памяти. И если экземпляр станет недоступным в процессе выполнения саги, то данная сага будет утеряна, а для приведения системы в согласованное состояние понадобится ручное вмешательство администратора.

Если реализовать модуль, контролирующий репликацию данных между оркестраторами, то этот модуль сам будет единой точкой отказа. Существует семейство алгоритмов Paxos, позволяющих достигать консенсуса состояний между репликами, рассмотрим одну из его реализаций.

3. Алгоритм Raft

Raft — это алгоритм консенсуса, предназначенный для поддержания одинакового состояния между реплицированными сервисами в распределенной системе. Как и Paxos, Raft гарантирует, что одновременно активен только один лидер и что все последователи в конечном итоге получат все подтвержденные записи даже в случае сбоев или разделения сети.

Алгоритм Raft состоит из трех ключевых ролей: лидера, последователя и кандидата. В кластере Raft одна реплика избирается лидером, а все остальные реплики являются последователями. Если лидер терпит неудачу, из кандидатов избирается новый лидер.

Лидер обрабатывает запросы и сообщает о своем состоянии или изменении состояния последователям. На рис. 4 представлен пример согласования состояний реплик методом Raft.

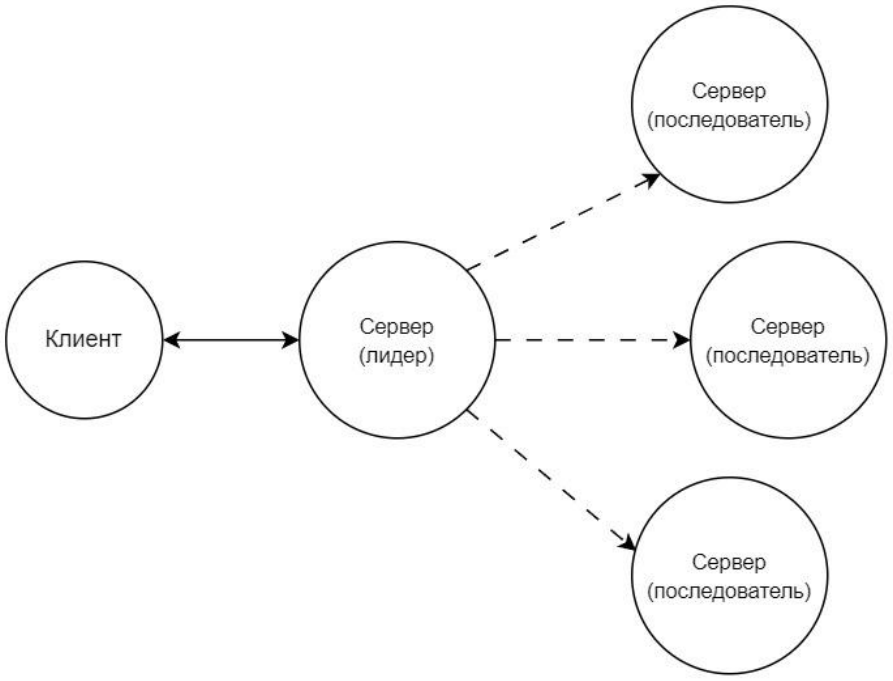


Рис. 4. Обработка запроса и согласование состояний алгоритмом Raft

Применив данный алгоритм к репликам оркестратора саги, возможно получить высокий уровень отказоустойчивости оркестратора и возможности системы обрабатывать запросы при отказе оркестратора. Такая система будет находиться в согласованном состоянии, будет иметь высокий уровень отказоустойчивости.

Для получения данных качеств системы необходимо предусмотреть в реализации оркестратора саги возможность создания саги с уникальным идентификатором, регистрацию компенсационного действия для саги, инициирование выполнения компенсационных действий по запросу, либо по истечении определенного времени без регистрации последующего сервиса. Так же оркестратор должен иметь методы для согласования состояния и выбора лидера алгоритмом Raft.

Заключение

В статье была рассмотрена проблема согласованности данных в распределенной системе, приведен способ решения шаблоном проектирования Saga. Рассмотрены варианты

реализации данного шаблона, а также проблема отказа оркестратора. Было рассмотрено решение с помощью репликации оркестраторов и согласования их состояния алгоритмом Raft.

Литература

1. Бёрнс Б. Распределенные системы. Паттерны проектирования / Б. Бёрнс – Санкт-Петербург : Питер, 2019. – 224 с.
2. Ньюмен С. Создание микросервисов. / С. Ньюмен. – Санкт-Петербург : Питер, 2016. – 304 с.
3. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга / К. Ричардсон – Санкт-Петербург : Питер, 2022. – 544 с.
4. Enes V. State-machine replication for planet-scale systems / // EuroSys 20: Fifteenth EuroSys Conference 2020 – Heraklion, Greece, 27–30 April 2020. – P. 1–15.
5. Jia Y. Adaptive Erasure Coded Data Maintenance for Consensus in Distributed Networks / Y. Jia, G. Xu, C. W. Sung, S. Mostafa // 2021 40th International Symposium on Reliable Distributed Systems (SRDS) – Chicago, USA, 20–23 September 2021. – P. 345–346.
6. Ongaro D. In Search of an Understandable Consensus Algorithm / D. Ongaro, J. Ousterhout // 2014 USENIX Annual Technical Conference – Philadelphia, USA, 19–20 June 2014. – P. 305–319.

УДК 004

ОПТИМИЗАЦИЯ ВЫЧИСЛЕНИЯ ПОЛОЖЕНИЯ ВИЗУАЛЬНЫХ ЭЛЕМЕНТОВ В IOS ПРИЛОЖЕНИИ С ИСПОЛЬЗОВАНИЕМ ОТДЕЛЬНОГО ПОТОКА

З. А. Сегал
Воронежский государственный университет

Введение

“Autolayout” и асинхронное вычисление фреймов – два разных подхода к управлению макетом и позиционированию элементов пользовательского интерфейса (UI) в разработке под iOS. В то время как Autolayout является стандартным способом управления макетом, асинхронное вычисление фреймов предлагает способ избежать блокировки главного потока и улучшить производительность. В этой статье мы сравним и сопоставим эти два подхода.

Сравнение Autolayout и асинхронного вычисления положения элементов

Autolayout – это система, основанная на ограничениях. Она позволяет разработчикам определять отношения между элементами пользовательского интерфейса. С помощью Autolayout разработчики могут создавать правила, которые определяют, как должны быть расположены и масштабированы элементы пользовательского интерфейса относительно друг друга. Autolayout гарантирует, что элементы пользовательского интерфейса будут выглядеть корректно на разных размерах экранов и в разных ориентациях. Autolayout вычисляет размер и положение каждого элемента пользовательского интерфейса на основе заданных ограничений. Autolayout работает синхронно и выполняется в главном потоке, что может сказаться на производительности.

С другой стороны, асинхронное вычисление положения элементов(фреймов) – это подход, при котором расположение элементов пользовательского интерфейса вычисляется асинхронно, на фоновом потоке, а не в главном потоке. С помощью этого подхода разработчики могут избежать блокировки главного потока, что крайне важно для поддержания плавности пользовательского опыта. Этот подход предполагает установку

начального размера элементов пользовательского интерфейса на значение по умолчанию, а затем асинхронное вычисление фактического размера. Асинхронное вычисление фреймов может улучшить производительность, особенно для сложных макетов, но требует дополнительных усилий при разработке.

Одним из преимуществ Autolayout является то, что он проще в использовании. Разработчики могут создавать ограничения в Interface Builder или в коде, а Autolayout

самостоятельно заботится об остальном. Autolayout также может адаптироваться к изменениям макета во время выполнения приложения, что делает его более гибким.

Autolayout может стать проблемой при работе с очень сложными макетами. Вычисление макета может занять значительное время и замедлить работу приложения. В таких случаях асинхронное вычисление фреймов может быть более эффективным вариантом. Однако, использование данного метода требует большего усилия при разработке.

Следует отметить, что каждый подход имеет свои преимущества и недостатки, и выбор между Autolayout и асинхронным вычислением фреймов зависит от конкретных требований проекта и ограничений, с которыми сталкиваются разработчики.

В целом, Autolayout более гибкий и адаптивный подход, который позволяет создавать макеты, которые автоматически адаптируются к различным экранам и устройствам. Это делает его особенно полезным для мобильных приложений, где устройства имеют различные размеры и разрешения экранов. Кроме того, Autolayout позволяет создавать сложные макеты с несколькими уровнями вложенности и ограничениями, что может быть трудно достичь с помощью асинхронного вычисления фреймов.

С другой стороны, асинхронное вычисление фреймов может быть более производительным подходом, поскольку он не использует систему ограничений и автоматически вычисляет координаты и размеры элементов пользовательского интерфейса. Это может предотвратить задержки в отображении элементов пользовательского интерфейса, особенно если приложение имеет большое количество элементов.

Но стоит отметить, что асинхронное вычисление фреймов имеет свои недостатки. В частности, такой способ может быть менее гибким и адаптивным, поскольку разработчикам приходится самостоятельно вычислять координаты и размеры элементов пользовательского интерфейса, что может быть сложно, особенно для перегруженных макетов. Кроме того, данный подход может быть более трудным для отладки и обслуживания, поскольку требует более низкоуровневого подхода к управлению макетом.

В итоге, выбор между Autolayout и асинхронным вычислением фреймов зависит от конкретных требований проекта и ограничений, с которыми сталкиваются разработчики. Если проект имеет сложный макет с несколькими уровнями вложенности и ограничениями, то Autolayout может быть более подходящим подходом. С другой стороны, если производительность является первостепенной задачей, то асинхронное вычисление фреймов может быть более подходящим вариантом.

В любом случае, разработчики должны тщательно взвешивать все преимущества и недостатки каждого подхода и выбирать тот, который наилучше подходит для поставленной задачи.

Заключение

Выбор между Autolayout и асинхронном вычислением фреймов зависит от конкретных потребностей проекта. Если в разработке используется сложный макет и не требуется оптимизация производительности, то Autolayout может быть идеальным выбором. Однако, если используется очень сложный макет и нужна максимальная производительность, то асинхронное вычисление фреймов может быть лучшим выбором. В любом случае, оба подхода имеют свои преимущества и недостатки, и выбор должен быть основан на конкретных потребностях проекта.

Список литературы

1. Neuburg M. Programming iOS 14 : Dive Deep into Views, View Controllers, and Frameworks, 2020 – ISBN 9781492092124, 1492092126.

2. Eidhof C. Thinking in SwiftUI: Updated for iOS 14, 2020 – ISBN 9798626292411.

3. Jeroen L. UIKit vs. SwiftUI: How to Choose the Right Framework for Your App. – Режим доступа: https://getstream.io/blog/uikit-vs-swiftui/. – (Дата обращения: 10.04.2023)

4. Apple developer documentation – Режим доступа: https://developer.apple.com/documentation/ – (Дата обращения: 05.04.2023).

УДК 004.94

ПОСТРОЕНИЕ ИГРОВОЙ АРХИТЕКТУРЫ UNITY3D, ОСНОВАННОЙ НА SCRIPTABLEOBJECTS

А. В. Семенов, Е. В. Трофименко

Воронежский государственный университет

Введение

Архитектура компьютерной игры (движка игры) — структура игры, как правило, включающая программные компоненты и взаимосвязи между ними. Архитектура игры определяет из каких элементов состоит игра и как эти элементы взаимодействуют между собой, но при этом описание не обязано быть исчерпывающим, достаточно бывает описания главных структурных элементов, элементов, связанных с основным поведением и элементов, которые определяют значимые свойства. Любая игра имеет архитектуру независимо от того, проектировалась она или нет. Архитектура может быть восстановлена по уже имеющейся игре. При создании игры не обязательно прорабатывать ее архитектуру заранее, однако ее проработка может существенно уменьшить объем работ при реализации игры, так как исправление архитектуры в общем случае проще исправления кода реализации. Описание архитектуры позволяет скрыть сложность игры при помощи абстракции и разделения ответственности между подсистемами. Наличие описания архитектуры, как правило, упрощает обсуждение и принятие решений, касающихся разрабатываемой игры.

1. Архитектура на основе ScriptableObjects

ScriptableObjects представляют из себя структуру хранения данных любого размера и сложности, использующую не только стандартные C# библиотеки, типы и методы, но и созданные специально под Unity. Они предоставляют механизм для динамической загрузки, обработки изменения и хранения данных. В конечном итоге они встраиваются в готовый продукт и хранятся вместе с скомпилированными игровыми файлами. Одной из особенностей Scriptable Object является глобальность изменений: у них нет как таковых экземпляров, изменение поля Scriptable Object файла затронет все компоненты, которые считывают из него данные.

При этом они легко адаптируются под разные нужды, и чтобы их создать или изменить не нужно открывать какой-либо внешний редактор - все изменения производятся внутри Unity-инспектора. Рассмотрим это подробнее на примере карточной игры.

2. Реализация архитектуры с использование ScriptableObjects на примере создания карточной игры

Создается карточная компьютерная игра. Цель игры - дойти до конца уровня. Уровень состоит из комнат в каждой из которых могут находится либо дополнительные карты для

игрока, либо враг. Враг каждый ход совершает действие - усиление себя, ослабление игрока, атака или защита. Задача игрока - победить врага путем разыгрывания карт, исходя из ситуации. У каждой карты есть цена розыгрыша, название, описание, картинка, и эффект (урон или защита). Тогда базовый ScriptableObject класс может выглядеть следующим образом:

```
using UnityEngine;
using UnityEngine.UI;
[CreateAssetMenu(fileName = "CardBase", menuName = "Scriptable Objects/Cards/CardBase")]

public class Card_SO : ScriptableObject
{
   public int cardCost;
   public string cardName;
   public Image cardImage;
   public string cardDescription;
}
```

Затем есть два варианта дальнейшей построения архитектуры проекта (рис. 1): расширять уже имеющийся класс, добавляя поля для различных значений по мере необходимости или создать несколько классов-наследников, расширяющих базовую функциональность для реализации других типов:

```
public class AttackCard_SO : Card_SO
{
      public int damageValue;
}
public class DefenseCard_SO : Card_SO
{
      public int defenseValue;
}
```

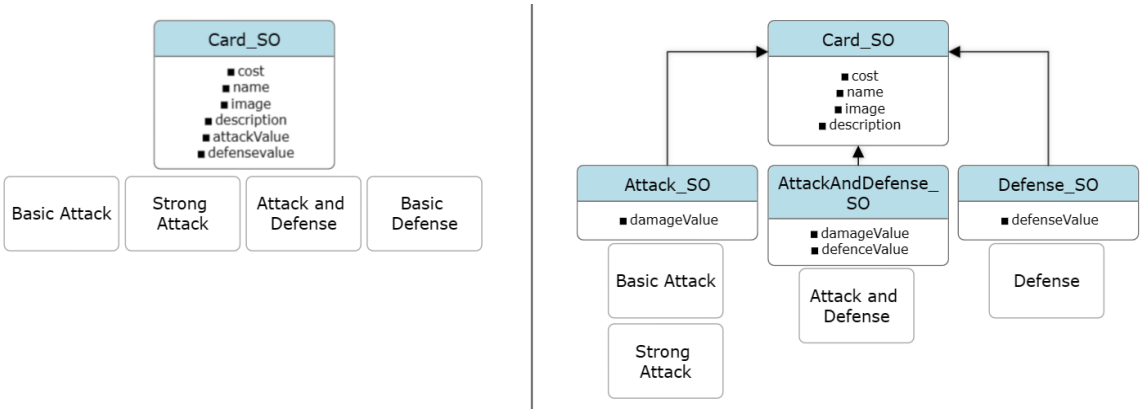


Рис.1. Пример диаграмм классов двух способов построения архитектуры

Оба подхода имеют свои преимущества и недостатки. В первом случае есть возможность делать разнообразные комплексные карты добавляя новые поля, но при этом всегда будет огромное количество неиспользуемых значений, растущее по мере усложнения проекта. Во втором случае получается более строгая и легко читаемая система, но при добавлении новых, карт может потребоваться создание новых типов. Уже в данном примере,

если карта будет как наносить урон, так и давать игроку защиту - придется создавать новый класс AttackAndDefenseCard_SO, в будущем, наверняка захочется иметь карты различных эффектов (кровотечение, отравление, получение дополнительной энергии), для которых тоже придется создавать новых наследников. Гейм-дизайнер, учитывая амбиции проекта, преимущества и недостатки каждого из подходов должен решить какой из способов более уместен в контексте разрабатываемого проекта.

При этом используя для хранения карт не MonoBehaviour или стандартные C# классы, а ScriptableObjects мы не теряем преимущества, которые они предоставляют. Всю необходимую логику все еще можно хранить внутри ScriptableObjects.

Предположим, что был выбран второй подход, тогда в базовый класс Card_SO можно добавить базовые виртуальные методы - действие при розыгрыше карты и получение ее подробного описания:

```
public virtual void PlayCard(Player _player, Enemy _enemy){}
public virtual string GenerateDescription() {}
```

А в классе-наследнике переписать их под нужды конкретного класса. Рассмотрим на примере создания сложной карты, наносящей урон игроку при использовании, восполняющей его энергию, наносящей урон противнику несколько раз и накладывающей на него эффект яда:

```
public class ComplexCard_SO : Card_SO
{
   public int damageEnemy;
   public int damageInstances;
   public int damageSelf;
   public int energyRestore;
   public int poisonDamage;

   public virtual void PlayCard(Player _player, Enemy _enemy)
   {
            _player.ApplyDamage(damageSelf);
      _player.RestoreEnergy(energyRestore);
            _enemy.ApplyDamage(damageEnemy, damageInstances);
            _enemy.ApplyPoison(poisonDamage);
   }

   public override string GenerateDescription()
   {
      string description = "Complex Card: ";
      description += "Deals " + damageSelf + " damage to the player. ";
      description += "Deals " + damageEnemy + " damage to enemy " +
                     " over " + damageInstances + " instances.”;
      description += "Restores " + energyRestore + " energy. ";
      description += "Applies " + poisonDamage + " poison to the enemy. ";
      return description;
   }
}
```

На рисунке 2 представлен пример диаграммы классов при реализации сложной карты.

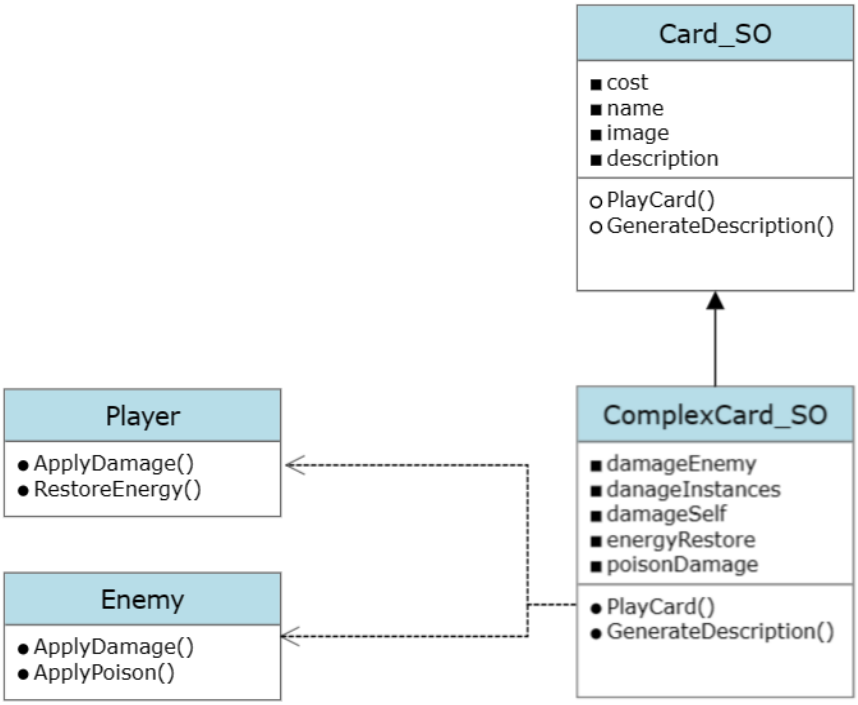


Рис.2. Пример реализации сложной карты

Таким образом, при отрисовке карты необходимо будет считать картинку, название и цену карты из полей ScriptableObject, а в качестве описания использовать строку из метода GenerateDescription(). Разыгрывая карту, достаточно будет вызвать метод PlayCard(), передав игрока и цель в качестве параметров.

Стоит заметить, что в данном случае ScriptableObjects так же применимы к случаю хранения информации о врагах, однако необходимо учитывать, что изменение поля health до нуля в одном “экземпляре” врага, на самом деле изменит глобальное значение внутри самого ScriptableObject, и у всех врагов, считывающих данные из него здоровье изначально будет равно нулю. В этом случае правильнее будет вынести логику в отдельный MonoBehaviour класс, у которого будет поле enemy_SO, хранящее ссылку на ScriptableObject из которого будут считываться значения необходимых полей (рис. 3).

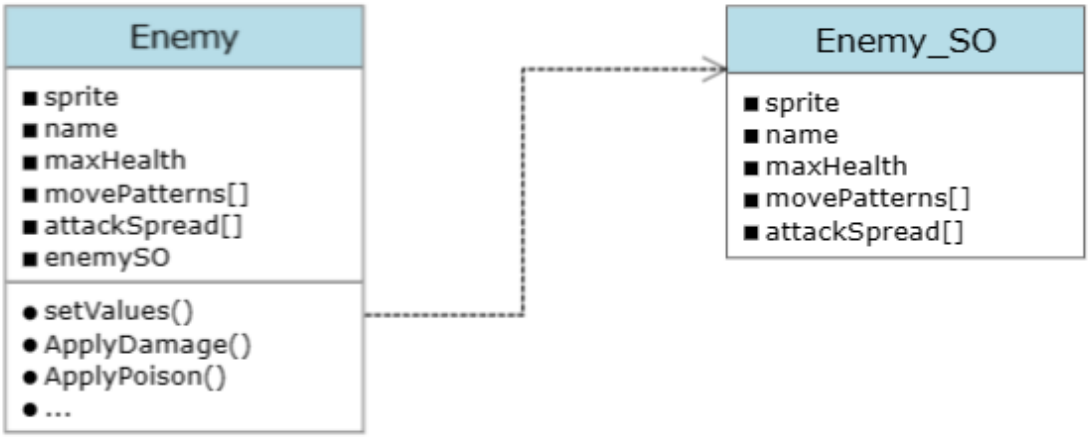


Рис.3. Диаграмма классов для реализации противников

Заключение

ScriptableObjects предоставляют базис для построения игровой архитектуры. Они являются Unity-ассетами - хранятся вместе с игровыми файлами - и их можно создавать и изменять прямо в редакторе. В готовый продукт они встраиваются, при этом поддерживают считывание и запись в реальном времени, а также могут содержать в себе игровую логику.

При правильном подходе, ScriptableObjects могут обеспечить модульную систему, растущую вширь, а не вглубь, что значительно повышает возможности кода и используемых сценариев.

Однако не стоит забывать, что изменение родительского, для игрового объекта, ScriptableObject отразится на всех объектах, считывающих из него данные.

При этом необходимо понимать, что ScriptableObjects - один из многих инструментов Unity разработчика и они подходят для конкретного пула задач (например, когда у нас множество похожих по структуре экземпляров одного родительского класса, как в примере с картами).

Литература

1. Unity User Manual 2022.2 [Электронный ресурс].-URL: https://docs.unity3d.com/Manual/index.html (Дата обращения: 29.03.2023)
2. Язык программирования C# и платформа .NET [Электронный ресурс].-URL: https://metanit.com/sharp (Дата обращения: 30.03.2023)
3. Бонд Д. Г. Unity и C#. Геймдев от идеи до реализации / Бонд Д. Г., Лемарчанд Р. - Санкт-Петербург: Питер, 2022 – 256 с.

УДК 004.4'2

АНАЛИЗ ФРЕЙМВОРКОВ ДЛЯ BACKEND-РАЗРАБОТКИ ПЛАТФОРМЫ ДЛЯ ТЕСТИРОВАНИЯ СТУДЕНТОВ

О. С. Сидоров

Воронежский государственный университет

Введение

Web-приложения – популярный тип программного обеспечения, работающий на основе клиент-серверной архитектуры. В работе проводится анализ возможностей некоторых фреймворков, используемых при написании серверной части приложения на языке JavaScript и значительно упрощающих работу при создании web-приложения. Важным критерием будет удобство использования инструментов при реализации приложения, а именно платформы для проведения тестов для контроля знаний обучающихся в школах и университетах, с расширенными возможностями по созданию вопросов и заданий. Данное приложение подразумевает ведение таблицы или таблиц различных типов вопросов, то есть для каждого типа вопроса могут быть заполнены различные поля, такие как картинка, варианты ответов, формулы и так далее. Также приложение должно содержать строгую типизацию серверной части.

Цель работы: проанализировать возможности инструментов по написанию серверной части на языке JavaScript, а также их применимость в рамках создаваемого приложения и способность поддерживать строгую типизацию.

Были поставлены задачи:

1. Найти возможные инструменты для написания серверной части web-приложения.
2. Проанализировать фреймворки, найти их преимущества и недостатки.
3. Сравнить их и выбрать подходящий для написания web-приложения.

1. Возможные инструменты

Самыми популярными на данный момент являются фреймворки Express, Nest. Фреймворки firebase, Meteor js не являются популярными, тем не менее стоит рассмотреть непопулярные варианты для построения более полной картины.

2. Анализ инструментов

2.1. Express

Express — это платформа веб-приложений для Node.js, предназначенная для создания веб-приложений и API с использованием SQL запросов.

Преимущества:

1. Простота в освоении и реализации.
2. Поскольку Express является базовой платформой, команда разработчиков Express разработала пакеты промежуточного программного обеспечения для решения различных проблем разработки.
3. Работает с SQL запросами, что упрощает работу.

Недостатки:

1. Сообщения об ошибках неисчерпывающие.
2. Могут быть проблемы с безопасностью.
3. Сложность создания проекта и работы с ним при подключении строгой типизации.

2.2. Nest

Nest — это платформа для создания программ Node.js на стороне сервера. Он создан и полностью поддерживает TypeScript.

Преимущества:

1. Встроенная типизация.
2. Исчерпывающая документация
3. Модульная структура, использование которой упрощает разделение проекта на отдельные блоки.
4. Много встроенных методов, что упрощает работу и одновременно усложняет поиск ошибок.

Недостатки:

1. Сложность работы с фреймворком, обязующая постоянно искать в документации информацию.
2. Запросы устроены своим, внутренним способом.

2.3. Meteor

Meteor — веб-платформа на языке JavaScript, предназначенная для разработки web-приложений реального времени.

Преимущества:

1. Встроенный канал передачи (WebSocket + DDP).
2. Встроенная база данных — MongoDB.

Недостатки:

1. Встроенная база данных.
2. Усложняет написание клиентской части нестандартными решениями.
3. Убирает разделение клиент-сервер.
4. Малое количество и излишнее усложнение документации.

2.4. Firebase

Firebase – облачная база данных не использующая SQL, поддерживаемая Google.

Преимущества:

1. Приложения, основанные на firebase работают без серверной части, все манипуляции проводятся на клиентской, тем не менее не перегружают её.
2. Встроенные инструменты авторизации, развёртывания, хранения.
3. Нет необходимости конструировать базу данных.
4. Хранение разных структур в одной коллекции.
5. Удобная и простая документация.

Недостатки:

1. Ограниченные возможности запросов.
2. Хранение разных структур в одной коллекции.
3. Запросы формируются на frontend, что усложняет его написание.

3. Сравнение инструментов

После анализа преимуществ и недостатков, можно дать краткую характеристику каждого из инструментов. Express - простой фреймворк, работающий с SQL, который практически не поддерживает типизацию, что усложняет разработку. Nest - обладает сложным, не похожим на стандартный JavaScript синтаксисом, также тот факт, что взаимодействие с базой данных происходит через внутренние команды, усложняет работу с базой данных. Meteor имеет плохую документацию, а также сложную структуру, что усложняет работу с ним. Firebase крайне прост в освоении, а хранение различных данных в одной коллекции, которое обычно является недостатком, в рамках требований является большим преимуществом. На основании данного анализа инструментов для серверной части был выбран Firebase, его простота изучения и удобство использования были важными характеристиками, кроме того, он наиболее подходит для реализации требований к приложению.

Заключение

В результате исследования были проанализированы инструменты для написания серверной части web-приложения. Были выявлены преимущества и недостатки данных инструментов, и на их основе выбран наиболее подходящий для написания приложения.

Литература

1. 9 Best JavaScript Frameworks to Use in 2023. – Режим доступа: https://ninetailed.io/blog/best-javascript-frameworks/ (Дата обращения: 19.04.2023).

2. Node.js и JavaScript для серверной разработки. – Режим доступа: https://habr.com/ru/companies/ruvds/articles/345164/ (Дата обращения: 19.04.2023).

3. Should you use NestJS for your next project? – Режим доступа: https://www.vairix.com/tech-blog/should-you-use-nestjs (Дата обращения: 19.04.2023).

4. Advantages and disadvantages of NestJS – Режим доступа: https://medium.com/mobile-reality/advantages-and-disadvantages-of-nestjs-76bcb60f5d63 (Дата обращения: 19.04.2023).

5. Боевой полет на Meteor-е – https://habr.com/ru/articles/437166/ (Дата обращения: 19.04.2023).

6. Firebase Pros and Cons: When You Should and Shouldn’t Use Firebase | OSDB – Режим доступа: https://osdb.io/firebase-pros-and-cons-when-you-should-and-shouldnt-use-firebase-osdb/ (Дата обращения: 19.04.2023).

7. A Quick Guide to MeteorJS – What it Is, and Who Should Use it – Режим доступа: https://www.freecodecamp.org/news/what-is-meteorjs-and-who-should-use-it/ (Дата обращения: 19.04.2023).

УДК 004

РАЗРАБОТКА КОРПОРАТИВНОГО МЕССЕНДЖЕРА

Е. М. Сухоруков

Воронежский государственный университет

Введение

Современное общество невозможно представить без средств связи. Большую часть времени люди проводят, используя различные гаджеты. Мессенджер – программное обеспечение, с помощью которого два пользователя (или группа пользователей) могут обмениваться текстовыми сообщениями или любой другой информацией, представленной в альтернативном варианте, в реальном времени.

Несомненно, огромным преимуществом мессенджеров является хранение сообщений, возможность в любой момент можно найти необходимую информацию.

Не все мессенджеры дают достаточную гарантию безопасности канала связи. Известно много случаев, когда находили уязвимость в приложении и происходила утечка данных.

Таким образом, задача шифрования канала связи является актуальной. Чтобы обеспечить высокий уровень шифрования, необходимо разработать собственный протокол связи, чтобы никто не смог получить информацию, передающуюся на сервер или на клиента.

В большинстве уже существующих подобных приложений некоторые функции можно использовать только если купить определенную подписку. В разработанном приложении не предусматривается никаких платных подписок, весь контент будет доступен бесплатно. Также на идею создания приложения повлияла частичная блокировка одного из популярных мессенджеров.

В большинстве существующих на данный момент мессенджерах очень много лишней функциональности, что зачастую мешает общению (реклама, огромное количество ненужных смайлов, эмодзи и т.п.). Из-за постоянного спама, важная информация теряется в чате и ее становится трудно найти.

Ранее компании использовали локальные чаты, то есть соединенные устройства не имели доступа в интернет и работали во внутренней сети. Такие приложения востребованы до сих пор, так как считаются достаточно безопасным способом обмена информацией. Отсутствие возможности подключения к удаленным серверам позволяет предотвратить попадание данных в руки третьих лиц.

При создании собственного корпоративного мессенджера появляется возможность ограничить круг лиц, которые могут им пользоваться, оптимизировать функциональность, что позволит комфортно общаться по рабочим вопросам, а также создать собственный протокол связи для клиент-серверного взаимодействия для обеспечения высокой гарантии безопасности данных.

1. Анализ существующих решений

Telegram — это кроссплатформенное приложение для мгновенной передачи сообщений. Программа написана на языке высокого уровня C. Используется протокол *MTProto*, который применяет многоуровневое шифрование. При авторизации пользователя применяются алгоритмы *RSA-2048, DH-2048*. Также используются криптографические хеш-алгоритмы *SHA-1* и *MD5*.

Особенность приложения в том, что при авторизации на новом устройстве не нужно загружать бэкап-файлы, все происходит автоматически, вся история загружается сама по себе, а файлы хранятся на стороннем сервере и доступны в любой момент.

Минус приложения в том, что некоторые функции, такие как отправка файлов больших размеров, скачивание медиафайлов без ограничения скорости и др. доступны только после покупки подписки «Telegram Premium».

WhatsApp — популярная бесплатная система мгновенного обмена текстовыми сообщениями для мобильных и иных платформ с поддержкой голосовой и видеосвязи. Программа была разработана на языке *Erlang*.

Является самым массовым мессенджером. WhatsApp использует модифицированный протокол *Extensible Messaging and Presence Protocol (XMPP)*. При установке создаётся аккаунт на сервере *S.whatsapp.net*, использующий номер телефона в качестве имени пользователя. Версия под *Android* автоматически применяет в качестве пароля *MD5*-хеш от изменённого идентификатора *IMEI*, а версия под *iOS* использует *MD5*-хеш от *MAC*-адреса.

Приложение автоматически синхронизирует список контактов с телефонной книгой устройства. Это возможно благодаря тому что все пользователи регистрируются при помощи номера телефона.

В мобильной версии *WhatsApp* прикрепленные к сообщениям медиафайлы автоматически сохраняется в галерею телефона. Это неудобно.

В последнее время часто появляется информация об утечке данных у пользователей WhatsApp.

Viber является кроссплатформенной программой, позволяющей совершать звонки и обмен сообщениями через *WiFi* и мобильную сеть. Также существует платная функция *ViberOut*, которая помогает связаться с устройством без доступа в интернет или же на котором не установлена программа. Приложение написано с помощью языков: *Java, C, Python, C++, Objective-C.*

Приложение имеет шифрование, секретные переписки, звонки. Добавляются видеозвонки. Есть множество стикеров как платных, так и бесплатных.

Viber можно считать корпоративным мессенджером, поскольку большие сети и магазины, у которых есть персональные данные в базе, могут присылать рекламу. С одной стороны, это очень удобно и практично, но с другой — надоедливо и многим не нужно.

В приложении также доступны многопользовательские чаты, боты и публичные каналы.

Связаться с кем-то из пользователей можно по номеру мобильного телефона, которые приложение синхронизирует с телефонной книги.

2. Реализация

В табл. представлены результаты анализа по тем критериям, которые важны для корпоративного мессенджера.

Критерии сравнения существующих решений

	Telegram	Viber	WhatsApp
Отображение статуса сообщений	Есть	Есть	Есть
Реклама	Мало	Много	Мало
Возможность отправлять медиафайлы больших объемов	Платно	Нет	Нет
Удаление сообщений	Полное удаление	Показывается, что сообщение удалено	Показывается, что сообщение удалено

На основе анализа существующих решений разработанный корпоративный мессенджер

получил следующую функциональность:

- регистрация и авторизация пользователя;
- обмен сообщениями;
- возможность прикреплять к сообщению медиафайлы;
- обеспечение высокой безопасности;
- создание групповых чатов;
- ответ на сообщения;
- возможность полностью удалять сообщения.

Авторизация реализована с помощью *JSON Web Tokens* (JWT). В простом понимании — это строка в специальном формате, которая содержит данные, например, ID и имя зарегистрированного пользователя. Она передается при каждом запросе на сервер, когда необходимо идентифицировать автора запроса. Таким образом, неавторизированные пользователи не смогут взаимодействовать с сервером.

Для общения клиента и сервера был использован *WebSoket*. *WebSocket* — это протокол, который создает интерактивное соединение между клиентом и сервером для обмена данными в режиме реального времени [1]. В отличие от *HTTP* протокола, который построен на модели «запрос-ответ», *WebSoket* позволяют установить соединение между клиентом и сервером и сохранять его пока это необходимо. Это позволяет значительно ускорить обмен данными между пользователями и сервером. *WebSoket* основан на работе протокола *TCP*, что позволяет создавать приложения реального времени. В таких приложениях пользователь может получать или отправлять информацию сразу же после публикации без необходимости периодического опроса источника информации. Таким образом, приложение дает пользователям ощущение того, что действия происходят именно в данный момент, без малейших задержек.

Поскольку *TCP* протокол был написан собственноручно, это позволило обеспечить высокую безопасность данных, передаваемых с клиента на сервер и обратно, так как алгоритм кодирования и декодирования сообщений никому неизвестен.

Заключение

Разработанное приложение предоставляет пользователям возможность обмена сообщениями, позволяет прикреплять к сообщениям медиафайлы, обеспечивает высокую безопасность, поддерживает создание групповых чатов и возможность полностью удалять сообщения.

Особенность приложения в том, что клиентская часть написана с помощью фреймворка *Avalonia*. Поскольку фреймворк позволяет создавать кроссплатформенные приложения, разработанное приложение можно будет масштабировать до кроссплатформенного и пользоваться им на любом устройстве.

Литература

1. Веб-сокеты: боевое применение. [Электронный ресурс]. – URL: https://habr.com/post/162301/ (Дата обращения 19.04.2023).

УДК 004.4'41

ПРИНЦИПЫ РАБОТЫ МОДУЛЯ ТРАНСЛЯЦИИ КОДА ЯЗЫКА ПРОГРАММИРОВАНИЯ PL+ НА ЯЗЫКИ PL/SQL И JAVA ПРИ РАЗВЁРТЫВАНИИ АВТОМАТИЗИРОВАННЫХ БАНКОВСКИХ СИСТЕМ

Д. Р. Телков

Воронежский государственный университет

Введение

На сегодняшний день в ИТ-сегменте направлений развития нашей страны существует актуальная потребность в использовании отечественных цифровых продуктов. Особенно остро имеется необходимость применения качественного российского ПО в такой стратегически важной сфере деятельности как экономическая. Разрабатываемые отечественные автоматизированные банковские системы (АБС) должны отвечать современным запросам рынка и покрывать всю требуемую банковской системе функциональность с высокой степенью отказоустойчивости и гибкости. В статье предлагаются к рассмотрению принципы функционирования и реализации отечественной среды разработки автоматизированных банковских систем для финансовых организаций CFT Platform IDE в разрезе состава реализации одного из ключевых модулей работы системы — трансляции языка PL+ в пакеты СУБД PL/SQL и Java-код при создании дистрибутивов банковского ПО. Значимость модуля трансляции в общей системе обуславливается его положением «на стыке» прикладного кода, организации хранения данных и реализации их обработки с помощью сервера приложений [1].

1. Принцип работы модуля

Рассмотрим общий состав компонентов системы, процесс их взаимодействия и позицию рассматриваемого модуля в общей архитектуре приложения.

Разработка автоматизированных банковских систем начинается с описания структуры процессов финансовой организации, в которую происходит внедрение АБС, на языке PL+. Этот статически типизированный язык программирования, заимствующий некоторые понятия ООП и привносящий свои структурные компоненты, используемые для описания банковских моделей, был разработан группой компаний «ЦФТ» более 20 лет назад для написания прикладного кода финансовых систем. Прикладной код, описывающий банковские процессы, впоследствии транслируется на языки PL/SQL и Java. Это неотъемлемая и основополагающая часть разработки АБС, позволяющая за одну итерацию трансляции развернуть всю логику организации банковских данных, способы работы с ними с сохранением последующей возможности обновления модели этих данных, её реорганизации и дополнения. В PL/SQL происходит трансляция хранимых данных и их структур, а в Java классы компилируются средства обработки этих данных. Именно поэтому модуль трансляции может считаться одним из ключевых компонентов системы при реализации комплексных финансовых приложений в CFT Platform IDE.

2. Алгоритмы работы и состав реализации модуля трансляции

Исходный код модуля трансляции написан на языке программирования Java, модуль

состоит из нескольких java-классов, каждый из которых реализует логику трансляции отдельных элементов модели АБС. Процесс трансляции представляет собой синтаксический разбор исходного PL+ кода с созданием текстового представления прикладной логики на целевом языке.

Рассмотрим обобщённый алгоритм трансляции.

В момент, когда происходит операция деплоя (развёртывания описанной модели АБС на сервер), написанный прикладной код записывается в строки одной из служебных таблиц, после чего генерируется пакет PL+ путём объединения всех строк в один крупный текстовый блок. В полученный текст вставляются PL+ операторы для сохранения информации об изначальной структуре логических секций прикладного кода и другие вспомогательные синтаксические элементы работы с ним. Далее происходит вызов парсера PL+ — программной единицы, выполняющей синтаксический анализ прикладного кода. Синтаксический разбор происходит в процессе «чтения» парсером сформированного пакета PL+ в соответствии с грамматикой языка, при этом создаётся промежуточное представление (Intermediate representation) в виде дерева абстрактного синтаксиса (структура, используемая для древовидного представления исходного кода, в которой внутренние вершины сопоставлены с операторами языка программирования, а листья с соответствующими операндами [2]), представляющее собой входные данные для генераторов кода. Преимущество использования такой структуры заключается в возможности применения одного парсера на несколько генераторов, т.е. обработка одного и того же исходного кода позволяет генерировать несколько разных выходных языков.

Если в процессе синтаксического разбора обнаруживаются ошибки, информация о них переносится в системную таблицу для дальнейшей обработки, грамматика языка основывается на описании синтаксиса на языке программы yacc (Yet Another Compiler Compiler) и описании лексем (минимальной автономной смысловой единицы, которая может содержать ключевое слово языка, идентификатор или другой элемент синтаксиса, который может быть распознан анализатором) на языке программы генерации лексических анализаторов lex. Далее производится анализ сгенерированного на предыдущем шаге дерева промежуточного представления, при этом происходит идентификация именованных объектов исходного кода (имена дополняются ссылками на определения объектов в промежуточном представлении), проверка совместимости типов и дополнение узлов формируемого дерева.

Следующий этап — генерация. На нём происходит обход полученного дерева, в результате которого создаётся текстовое представление прикладной логики на целевом языке (PL/SQL, Java в зависимости от генератора), семантически эквивалентное исходному коду. Попутно происходит процесс оптимизации генерации за счёт применения метода оптимизации Constant folding, производящего замену константных выражений и переменных на их значения на этапе компиляции. На последнем шаге происходит сохранение полученных результатов, сгенерированный java-код записывается в системной таблице, для сгенерированного PL/SQL создаются и компилируются спецификации и тело пакетов для хранения на схеме, также в отдельной системной таблице сохраняется описание всех внешних ссылок, встретившихся в процессе компиляции. Эта информация используется для отслеживания зависимостей между методами. Если на одном из этапов произошли ошибки, не позволяющие продолжать компиляцию, то компиляция прерывается с сообщением соответствующего статуса пользователю.

Впоследствии сгенерированные PL/SQL пакеты классов (PL+ парадигме называемых «ТБП» — типами базовых понятий) описывают структуру хранения данных АБС, а сгенерированные Java-классы используются сервером приложений для их обработки.

Заключение

Были изложены принципы функционирования и алгоритм работы модуля трансляция кода языка программирования PL+ на языки PL/SQL и Java при развёртывании автоматизированных банковских систем, входящего в состав среды разработки CFT Platform IDE, которая является востребованной в отечественном сегменте. Модуль отвечает запросам современного рынка и позволяет решать прикладные задачи разработки отечественных АБС, покрывая всю требуемую банковской системе функциональность с высокой степенью отказоустойчивости и гибкости [3] (доцент, канд. техн. наук, доцент кафедры ПОиАИС ВГУ, Курченкова Татьяна Викторовна).

Литература

1. Документация Платформа 2 МСА – Режим доступа: https://nexus.cft.ru/dbi/2MCA.pdf – (дата обращения: 06.04.2023)
2. Abstract Syntax Tree | DeepSource – Режим доступа: https://deepsource.com/glossary/ast/ – (дата обращения: 04.04.2023)
3. Документация CFT Platform IDE – Режим доступа: https://www.cft.ru/sites/cft/Documents/CFT_ID%20Platform.pdf – (дата обращения: 04.04.2023)

УДК 519.8

МОДЕЛИ, АЛГОРИТМЫ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СОГЛАСОВАНИЯ ИНТЕРЕСОВ ЗАКАЗЧИКА И ИНВЕСТОРА ПРИ РЕАЛИЗАЦИИ ПРОЕКТА

Ю. А. Телкова, Д. А. Крыжко

Воронежский государственный университет

Введение

Деятельность каждой современной организации в той или иной степени связана с реализацией проектов. Проект представляет собой комплекс взаимосвязанных мероприятий, направленных на создание уникального продукта или услуги в рамках временных и ресурсных ограничений [1]. Традиционно в жизненном цикле проекта выделяется ряд этапов, включая концепцию, планирование, реализацию и завершение. Каждая стадия проекта предполагает привлечение групп специалистов определенной сферы и затрагивает интересы не только непосредственных его участников, но и иных стейкхолдеров, в чьих интересах успешное завершение данного проекта. В числе основных стейкхолдеров в общем случае выделяют инициатора проекта, куратора, заказчика, инвестора, руководителя проекта, команду проекта, генерального контрактора, генерального поставщика, консультанта, региональные органы власти, активные группы населения, экологические организации и т.д. [2]. Отметим, что каждый из стейкхолдеров имеет свои собственные цели и интересы, достижение которых является основным фактором участия в данном проекте. Поскольку интересы стейкхолдеров могут носить и противоречивый, конфликтный характер, то для успешной реализации проекта важным для его руководителя становится выявление, прогнозирование и согласования интересов всех его участников. Понимание и изучение вопросов согласования интересов участников проекта нашли свое отражение в трудах современных отечественных и зарубежных исследователей [3–5]. Обоснованность данному процессу придает использование логически верных процедур, подходов, алгоритмов и моделей теории принятия решений, теории игр и математического моделирования. В силу этого, разработка математического и программного инструментария, позволяющего оперативно и своевременно принять обоснованное решение по согласованию интересов стейкхолдеров проекта, является актуальной задачей.

Целью настоящего исследования является совершенствование процесса управления проектами посредством разработки математических моделей и алгоритмов согласования интересов заказчика и инвестора, позволяющих оказать поддержку в принятии решений по выбору варианта проекта и механизму финансирования.

1. Материалы и методы

Будем предполагать, что некоторая организация планирует реализовать проект, выступая при этом в роли инициатора и заказчика. Данную организацию будем в дальнейшем именовать заказчиком. В интересах заказчика – создание конкурентоспособного продукта, приносящего определенную прибыль.

Считаем, что по инициативе заказчика разработано N вариантов реализации проекта: $\tilde{A} = \{A_1, A_2, ..., A_N\}$. Различные варианты могут отличаться рядом характеристик – сроком окончания проекта, перечнем работ, стоимостью ресурсов, качеством выполнения работ,

риском превышения сроков, прибылью и т.д. Полагаем, что заказчиком определены K наиболее существенных характеристик, которые могут выступать в качестве критериев сравнения вариантов проекта и оценки их привлекательности для заказчика. Такие частные критерии обозначим через $F_1, F_2, ..., F_K$.

Для каждого варианта проекта A_n разработан комплексный укрупненный план, согласно которому рассчитана его длительность T_n, определены объемы финансовых средств Q_{nt}, необходимые для успешного проведения работ в каждый момент времени $t = 1, \ldots, T_n$. Временной интервал полагаем дискретным, а момент времени t может быть днем, неделей, месяцем, кварталом, годом и т.п.

Известно, что в каждый момент времени t (где $t = 1, \ldots, T$, $T = \max\limits_{1 \le n \le N} \{T_n\}$) для реализации проекта заказчик может выделить собственные средства в объеме, не превышающем величины $\overline{J_t}$. Суммарный объем собственных средств, который заказчик может вложить в проект, составляет $\tilde{J}$.

Предполагаем, что заказчик может упорядочить (проранжировать) варианты проекта $\tilde{A} = \{A_1, A_2, ..., A_N\}$ по предпочтению для реализации. Без ограничения общности будем считать, что:

$$A_1 \ge A_2 \ge \ldots \ge A_N, \tag{1}$$

где $A_i \ge A_j$ означает, что вариант проекта A_i для заказчика не хуже, чем A_j.

Ранжирование (1) может быть осуществлено на основании решения многокритериальной задачи:

$$\begin{gathered} F_1(A) \to \max, \\ F_2(A) \to \max, \\ \ldots \\ F_K(A) \to \max, \\ A \in \tilde{A}. \end{gathered} \tag{2}$$

Одним из методов решения задачи (2) является построение интегрального критерия

$$F(A) = F(F_1(A), ..., F_K(A)). \tag{3}$$

Одной из распространенных на практике форм представления интегрального критерия является взвешенная сумма предварительно нормализованных частных критериев:

$$F(A) = \alpha_1 . F_1^{norm}(A) + \alpha_2 F_2^{norm}(A) + \ldots + \alpha_K F_K^{norm}(A), \tag{4}$$

где $\alpha_1, \ldots, \alpha_K$ – веса важности критериев, обладающие следующими свойствами: $\sum\limits_{k=1}^{K} \alpha_k = 1$, $\alpha_k \ge 0$ для всех $k = 1, ..., K$. Веса важности назначаются экспертами или формируются на основе экспертных оценок методами обработки экспертной информации

Для формирования интегрального критерия и последующего упорядочения вариантов проекта могут быть выбраны и иные, более сложные процедуры принятия решений. В числе таких процедур – метода анализа иерархий Т. Саати [6].

Естественно, что при условии достаточности собственных средств, заказчик выберет для реализации наиболее предпочтительный вариант проекта A_1. При недостатке собственных средств заказчик должен найти инвестора, для которого данный проект может представлять финансовый интерес.

Основной целью инвестора в проекте является получение наибольшей прибыли путем

инвестирования в проект. Финансовые цели инвестора могут частично совпадать с целями заказчика, а могут быть и противоположными. Заказчик предлагает инвестору различные варианты проекта, для каждого из которых инвестор определяет целесообразность его реализации посредством подбора системы собственных и заемных платежей, обеспечивающих наибольшее значение чистого дисконтированного дохода (NPV проекта). Данная задача и составляет задачу инвестора. Перейдем к ее формальной постановке.

Рассмотрим вариант проекта A_k. Заказчик предоставляет инвестору документацию проекта и информацию о минимальных $\underline{J_t}$ и максимальных $\overline{J_t}$ размерах собственных финансовых средств, которые он может (или должен) вложить в проект в каждый момент времени *t*. Тогда финансирование заказчиком проекта J_t в момент времени t находится в диапазоне:

$$\underline{J_t} \le J_{nt} \le \overline{J_t}. \qquad (5)$$

Недостающие средства в объеме $\underline{I_{nt}} = Q_{nt} - J_{nt}$ (где $t = 1,...,T_n$) представляются инвестором.

Для финансирования проекта инвестор может задействовать как собственные, так и заемные средства. Объем собственных средств инвестора, вкладываемых в проект в момент *t* обозначим через переменную x_{nt}. При этом должно выполняться условие:

$$0 \le x_{nt} \le D_t, \quad t = 1,...,T_n, \qquad (6)$$

где D_t – максимально возможный размер финансирования за счет собственных средств.

Заемные средства инвестор может получить в финансовой организации в объеме y_{nt}, не превышающем величину $B_t(r)$:

$$0 \le y_{nt} \le B_t(r), \qquad t = 1,...,T_n, \qquad (7)$$

где r – ставка кредитования, удовлетворяющая условию: $r \ge r_{\min}$.

Тогда размер инвестируемых средств I_{nt} рассчитывается по формуле:

$$I_{nt} = x_{nt} + y_{nt}.$$

Прогнозируемое значение прибыли от реализации проекта в каждый момент времени *t* обозначим через $\tilde{V}_{nt}$. Из них инвестору отчисляется заданная доля β, где $0 \le \beta \le 1$. Тогда планируя финансовые расходы и выплаты, инвестор рассчитывает на денежные поступления в объеме $W_{nt} = \beta \cdot V_{nt}$.

Прибыль от реализации проекта может быть частично или полностью использована на погашение кредитного долга. Сумма погашения кредитного долга z_{nt} ограничена обязательной величиной и возможностями инвестора:

$$\underline{z_{nt}} \le z_{nt} \le W_{nt}. \qquad (8)$$

Полагаем, что неизрасходованные финансовые средства проекта аккумулируются и могут быть использованы в последующие моменты времени. Тогда в каждый момент времени *t* расходы по проекту составляют величину

$$I_{nt} = J_{nt} + x_{nt} + y_{nt} - z_{nt} + R_{nt}, \qquad (9)$$

где R_{nt} – неизрасходованные средства предыдущих периодов:

$$R_{nt} = I_{nt-1} - Q_{nt-1}, \; t = 1,...,T_n,$$
$$R_{n0} = 0.$$

При этом проект должен быть полностью обеспечен финансами:

$$I_{nt} \ge Q_{nt}, \qquad t = 1,...,T_n. \qquad (10)$$

Задача инвестора – найти такие значения $\{x_{nt}, y_{nt}, z_{nt}\}_{t=1,\dots,T}$, которые удовлетворяют условиям (6)-(10) и обеспечивают получение максимальной величины NPV проекта:

$$NPV = \sum_{t=1}^{T_n} \frac{W_{nt} - x_{nt} - z_{nt}}{(1+d)^t} \to \max, \qquad (11)$$

где d – ставка дисконтирования.

Задача инвестора (5)-(10) представляет задачу линейного программирования и может быть решена симплекс-методом.

Пусть NPV_n^* – оптимальное значение функции цели (11). Инвестор принимает проект, если:

$$NPV_n^* \geq \underline{NPV}, \qquad (12)$$

где NPV – установленное инвестором пороговое значение.

Согласование интересов заказчика и инвестора в таких условиях происходит по следующему *алгоритму*:

Шаг 1. Заказчик упорядочивает по предпочтительности варианты проекта (1).

Полагаем $n=1$.

Шаг 2. Заказчик предлагает к рассмотрению инвестора проект n. Инвестор решает задачу (5)-(10) и определяет целесообразность инвестиций по критерию (12).

Если вариант принимается, то останов.

Шаг 3. Полагаем $n=n+1$. Если $n \leq N$, то переход к шагу 2. Иначе – останов, инвестор не соглашается на реализацию проекта.

Заметим, что в представленной алгоритмом схеме взаимодействия заказчик и инвестор самостоятельно определяют размеры платежей. Вместе с тем в интересах и заказчика, и инвестора получение наибольшей прибыли от проекта. Повышение прибыли может быть обеспечено посредством совместного планирования платежей по проекту. В этом случае решается задача *согласованного выбора варианта проекта*.

Задача согласованного выбора варианта проекта формально представим как многокритериальную задачу оптимизации с критериями максимизации NPV для инвестора и заказчика:

$$NPV_1 = \sum_{t=1}^{T_n} \frac{W_{nt} - x_{nt} - z_{nt}}{(1+d)^t} \to \max,$$

$$NPV_2 = \sum_{t=1}^{T_n} \frac{(1-\beta)V_{nt} - J_{nt}}{(1+d)^t} \to \max, \qquad (13)$$

и системой ограничений (5)–(10).

Переменными модели (13) выступают $\{J_{nt}, x_{nt}, y_{nt}, z_{nt}\}_{t=1,\dots,T}$, а ее решение сводится к нахождению компромиссной Парето-оптимальной точки. Для практической реализации данной задачи выбран метод взвешенных сумм.

2. Практическая реализация

Для практической реализации алгоритма согласования интересов заказчика и инвестора разработан программный продукт. Программа и весь интерфейс написаны на языке C++ в VisualStudio 2022 года с использованием платформы .NET Framework версии 4.7.2.

В файле investor.txt представлены входные данные для модели (6)–(11): количество периодов, размер ставки дисконтирования и объем собственных средств инвестора, которые им могут быть вложены в проект в период *t*. Иллюстрация данных из входного файла investor.txt

представлена на рис. 1.

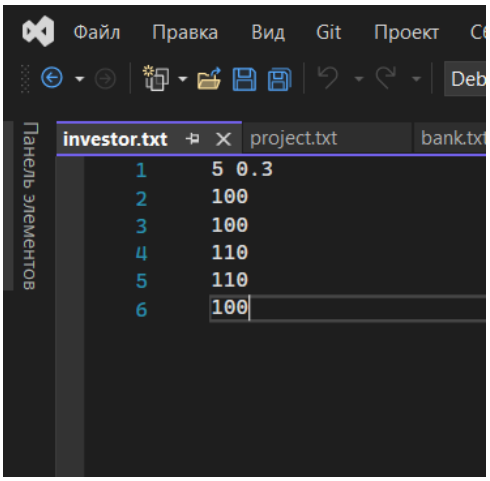


Рис. 1. Входные данные из файла investor.txt

В файле bank.txt находятся максимально возможная сумма кредитования в период t по ставке *r* и планируемый объем погашения кредита. Иллюстрация данных из входного файла файле bank.txt представлена на рис. 2.

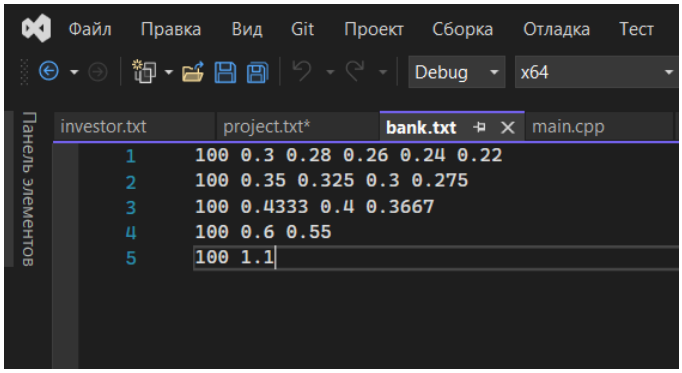


Рис. 2. Входные данные из файла bank.txt

Результаты работы программного кода представлены на рис. 3.

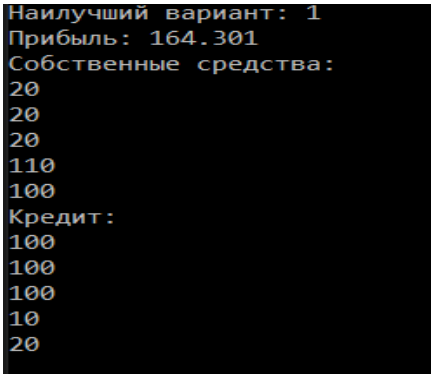


Рис. 3. Результаты работы программы

По результатам расчета NPV проекта для инвестора составляет 164.301 условную денежную единицу. При условии, что пороговое значение для данного показателя составляет 130 денежных единиц, инвестором принимается первый вариант проекта.

Заключение

Настоящая статья посвящена разработке математического и программного инструментария решения актуальной задачи в области управления проектами – согласования интересов заказчика и инвестора. Сформированы математическая модель выбора варианта проекта заказчиком и модель расчета оптимальных инвестиций инвестора (задача инвестора). Представленные модели составили основу алгоритма согласования интересов заказчика и инвестора. В работе представлен многокритериальный вариант модели, когда заказчик и инвестор совместно определяют финансовые потоки, что обеспечивает гибкое финансирование и, как следствие, увеличение прибыли. Дальнейшая работа в данном направлении предполагает усовершенствование программного обеспечения и проведение расчетов на данных ИТ-проекта.

Литература

1. Азбука управления проектами / Т.А. Аверина, С.А. Баркалов, Е.В. Баутина [и др]. – Старый Оскол : ТНТ, 2018. – 328 с.
2. Полковников А.В. Управление проектами. Полный курс MBA / А.В. Полковников, М.Ф. Дубовик. – М.: ЗАО «Олимп-Бизнес», 2015 – 552с.
3. Умное управление проектами / С.А. Баркалов, В.Н. Бурков, Я.Д. Гельруд [и др]. – Челябинск: Издательский Центр ЮУрГУ, 2019. – 189 с. – ISBN 978-5-696-05051-5.
4. Бондаренко Ю.В. Математическая и программная поддержка формирования календарного плана проектов с учетом вспомогательных ресурсов / Ю.В. Бондаренко, О.В. Бондаренко // Сб. тр. Международной конференции Актуальные проблемы прикладной математики, информатики и механики : сборник трудов Международной научной конференции, Воронеж, 13–15 декабря 2021 г. – Воронеж : Издательство «Научно-исследовательские публикации», 2022 – С. 1732-1738.
5. Каппелс Т. М. Финансово-ориентированное управление проектами / Т. М. Каппелс – Москва : ЗАО «Олимп – Бизнес», 2008. – 400 с. – ISBN 978-5-9693-0083-5.
6. Ketankumar R.R. Multi Objective Multi Mode Project Management Problem in Triangular Fuzzy Environment / R. R. Ketankumar, J. M. Dhodiya // International Journal of Innovative Technology and Exploring Engineering (IJITEE). – 2019. – Т. 9. – № 2. – С. 1772–1780.
7. Модели и алгоритмы принятия решений : учебное пособие / Т.В. Азарнова [и др.]. – Воронеж : Издательский дом ВГУ, 2021. – 310 с.

УДК 004.42

РАСПРЕДЕЛЕННОЕ СЕТЕВОЕ ФАЙЛОВОЕ ХРАНИЛИЩЕ

В. С. Тройнин

Воронежский государственный университет

Введение

В современном мире люди испытывают острую потребность в безопасном хранении собственных данных. Есть несколько основных вариантов того, как это можно сделать:

1. Данные можно хранить на собственном устройстве. Это самый простой и удобный способ, однако неполадка устройства или его утрата могут повлечь за собой безвозвратную потерю данных.
2. Данные можно хранить в облачным хранилище. Это довольно удобный и надёжный вариант, но пользователь доподлинно не знает, каким образом шифруются его данные и кому они передаются, и, следовательно, не может в полной мере доверять подобному сервису. Не говоря о том, что в такой централизованной системе может случиться утечка данных.
3. Данные можно хранить в блокчейн-хранилище. Это самый надёжный, но при этом самый сложный способ хранения с точки зрения пользователя. Более того, разработчик подобной системы никак не сможет монетизировать собственную разработку.

Однако, можно ли совместить преимущества всех вышеописанных подходов? Можно, при построении хранилища на основе *Blackboard-архитектуры*, в котором сами пользователи будут выступать в качестве узлов хранения данных, а сам сервис будет лишь обеспечивать взаимодействие между ними. [1]

1. Постановка задачи

Необходимо спроектировать веб-приложение на основе *Blackboard-архитектуры*, позволяющего безопасно сохранять файлы с помощью децентрализованного хранилища из множества пользователей системы.

2. Общий алгоритм работы приложения

Приложение строится на основе клиент-серверной архитектуры. Взаимодействие между клиентом и сервером осуществляется посредством протокола HTTP.

2.1. Алгоритм сохранения данных клиентом

Алгоритм сохранения данных клиентом работает следующим образом:

1. Клиент для каждого сохраняемого файла задает уникальное имя, версию (начиная с 0-ой), количество блоков (размер одного блока равен 1 Мб) и ключ шифрования.
2. Клиент шифрует файл, а затем делит его на блоки заданного размера.

3. Для каждого блока клиент высчитывает идентификатор (хеш-сумма от идентификатора пользователя, имени файла и номера блока) и подпись (хеш-сумма от идентификатора блока, версии файла и зашифрованного блока данных).
4. Клиент отправляет каждый блок файла на сервер.
5. Сервер высчитывает адрес для каждого блока (хеш-сумма от идентификатора блока и номера копии).
6. Сервер осуществляет поиск активного клиента с наибольшим идентификатором, не превышающим рассчитанный адрес, и затем передаёт найденному клиенту зашифрованный блок данных с номером его версии и подписью.
7. Пункты 1–6 повторяются до тех пор, пока клиент не сохранит все блоки своего файла.

2.2. Алгоритм восстановления данных клиентом

Алгоритм восстановления данных клиентом обратно работает следующим образом:
1. Клиент запрашивает у сервера идентификатор блока последней версии.
2. Сервер осуществляет поиск всех копий блоков в системе и проверяет их подписи.
3. Если находится хотя бы один корректный блок, то сервер передаёт эти данные клиенту.
4. Пункты 1–3 повторяются до тех пор, пока клиент не получит все блоки файла, а затем расшифровывает его.

2.3. Экономическая модель системы

За предоставление серверу корректных блоков данных клиент получает вознаграждение в виде электронных баллов, которые он может потратить на аренду памяти для собственных данных. Также баллы можно купить напрямую у сервиса, что обеспечит монетизацию проекта.

3. Архитектура приложения

3.1. Архитектура серверной части

В качестве основного фреймворка разработки был выбран *Spring Framework,* позволяющий создавать высокопроизводительные серверные приложения. [6]

В качестве СУБД была выбрана *PostgreSQL*, так как это производительное надёжное решение с открытом исходным кодом. [2] Для объектно-реляционного отображения между серверным приложением и системой управления базами данных была использована библиотека *Hibernate*, являющаяся реализацией JPA-спецификации. [6]

Архитектура реализуемого серверного приложения соответствует классической архитектуре *Spring MVC* приложения, то есть состоит из трёх уровней: [6]
1. *Контроллеры* (с их помощью создаётся API приложения).
2. *Сервисы* (в них реализуется бизнес-логика приложения).
3. *Репозитории* (служат для взаимодействия с СУБД).

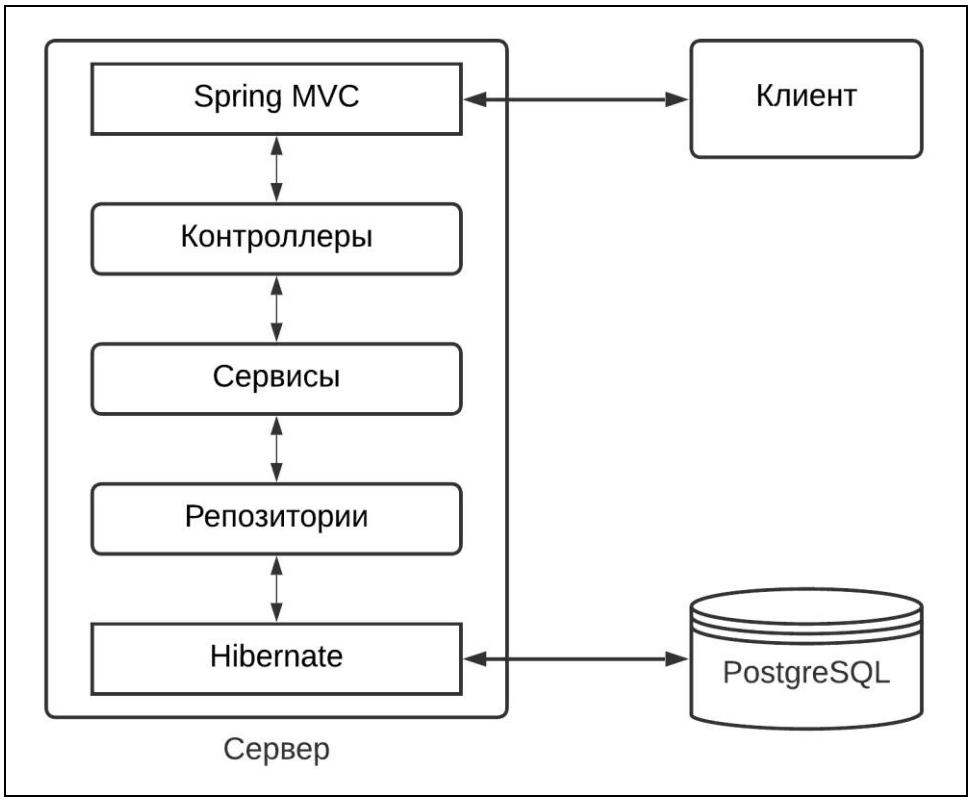


Рис. 1. Схема архитектуры серверной части веб-приложения

3.2. Архитектура клиентской части

В качестве основного фреймворка разработки был выбран *Swing*, позволяющий создавать кроссплатформенные десктоп-приложения. [5]

В качестве СУБД была выбрана SQLite, так как она встраивается в код как библиотека и не требует от пользователя отдельной установки. [4]

4. Программный интерфейс серверной части

В табл. 1 представлен перечень основных эндпоинтов и набор поддерживаемых для них HTTP-методов.

Таблица 1.

Краткая информация о поддерживаемом API сервера

Эндопоинт	**GET**	**POST**	**PUT**	**PATCH**	**DELETE**
/blocks	–	+	–	–	–
/blocks/{id}	+	–	+	+	+
/updates	+	–	–	–	–
/scores	+	–	–	–	–

Заключение

Было спроектировано веб-приложения на основе *Blackboard-архитектуры*, позволяющее безопасно сохранять файлы с помощью децентрализованного хранилища из множества пользователей системы.

Литература

1. Blackboard architecture: Documentation [Электронный ресурс]. URL: https://social.technet.microsoft.com/wiki/contents/articles/13215.blackboard-design-pattern.aspx (дата обращения: 15.04.2023).
2. PostgreSQL: Documentation [Электронный ресурс]. URL: https://www.postgresql.org/docs/ (дата обращения: 15.04.2023).
3. Spring: Documentation [Электронный ресурс]. URL: https://docs.spring.io/spring-framework/docs/current/reference/html/ (дата обращения: 15.04.2023).
4. SQLite: Documentation [Электронный ресурс]. URL: https://www.sqlite.org/docs.html (дата обращения: 15.04.2023).
5. Swing: Documentation [Электронный ресурс]. URL: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html
6. Spring 5 для профессионалов / Ю. Козмина, Р. Харроп, К. Шефер, Х. Кларенс — СПб.: ООО «Диалектика», 2020 — 1120 с.

УДК 004.042

АНАЛИЗ ПЛАТФОРМЫ APACHE KAFKA

Е.В. Трофимцов

Воронежский государственный университет

Введение

Apache Kafka – это распределенная платформа потоковой передачи событий с открытым исходным кодом, обеспечивающая высокую пропускную способность. Kafka была разработана в LinkedIn, после исходный код был передан в Apache Software Foundation. Написанная на Java и Scala, представляет собой шину сообщений системы Publish/Subscribe messaging, ориентированную на потоки и воспроизведение данных с высокой интенсивностью. Область применения кафки сфокусирована в отраслях, где требуются сбор, обработка, безопасное хранение и передача больших объемов данных.

1. Архитектура

Для начала рассмотрим основные сущности инструмента.

Broker – узел кластера, выполняющий следующие функции:

1. Получает сообщения от продюсеров, присваивает им смещения и отправляет их в дисковое хранилище.
2. Обслуживает консюмеры и отвечает на запросы выборки из партиций, возвращая записанные на диск сообщения.
3. Сохраняет информацию или в течение определенного промежутка времени или по достижению топиком определенного размера.

Cluster controller - один из брокеров, отвечающий за основные функции кластера:

- распределение партиций по брокерам;
- выбор ведущих реплик для партиций;
- мониторинг отказов.

Каждая партиция принадлежит одному из брокеров кластера или назначается нескольким брокерам, в результате чего произойдет репликация. Например, для leader партции, одна реплика из каждой партиции назначается ведущей, она выполняет все запросы чтобы обеспечить согласованность и знает какие ведомые ей реплики актуальны по сравнению с ней.

Followers партициями назначаются остальные реплики партиции. Они не обслуживают клиентские запросы, их задача реплицировать сообщения от ведущей реплики и поддерживать актуальное состояние. В случае аварийного сбоя ведущей реплики партиции одна из согласованных ведомых будет повышена в ранге и станет новой ведущей. Допустимые опции партиций для корректной работы брокеров:

- In sync - стабильно запрашивающая новые сообщения реплика
- Out of sync - отстающая на 10 секнуд реплика
- Preferred leader - предпочтительная ведущая реплика

Обработка запросов:

1. Acceptor Thread - для каждого порта на котором брокер выполняет прослушивание запускается принимающий поток.
2. Processor Thread - принимающий поток создает соединение и передающий контроль над ним обрабатывающему потоку.
3. Network Thread - поток обработчик отвечает за сбор запросов (request queue) от клиентов, сбор ответов (response queue) для клиентов.

Виды запросов:

1. Запросы от продюсеров - отправляются продюсерами и содержат сообщения, записываемые клиентами в брокеры.
2. Запросы на извлечение - отправляются консюмерами и ведомыми репликами при чтении сообщений от брокеров.
3. Запросы метаданных - включает список топиков интересующих клиента.

Основная единица физического хранения - реплика партиции. Распределение партиций происходит на основе заданных правил. Основная задача – равномерно распределить реплики по брокерам, реплики для каждой из партиций разместить на разных брокерах, реплики для каждой из партиций разместить на различных стойках.

Управление физическими файлами происходит отдельным сервисом. Партиции разбиваются на сегменты и каждый сегмент содержит один гигабайт данных или данные за определенный срок, в зависимости от того что оказывается меньше. По достижении этого лимита при записи брокером данных в партицию файл закрывается и начинается новый. Активным является сегмент, в который в настоящий момент производится запись.

Producers – сервисы, ответственные за генерацию новых сообщений и отправку их в топики или несколько топиков:

- Для генерации сообщений понадобится сначала создать объект ProducerRecord, включающий топик и значение (ключ и партиция необязательны).
- После отправки объекта ProducerRecord сериализует объекты ключа и значения записи в байтовые массивы для отправки по сети.
- Если в ProducerRecord была указана партциия, объект Partitioner возвращает ее, иначе выбирает партицию в соответствии с ключом.
- После этого запись помещается в пакет записей, предназначенных для отправки в соответствующие топик и партицию.
- В случае успешной записи сообщений брокер вернет объект RecordMetadata содержащий топик/партицию/смещение, иначе вернет ошибку.

Consumers – сервисы, которые подписываются на один топик или более и читают сообщения в порядке их создания.

Consumer group – один или несколько консюмеров, объединившихся для обработки партиций топика в группу:

- Организация в группы гарантирует чтение каждой партиции только одним членом группы.
- Возможность горизонтального масштабирования для чтения топика с большим количеством сообщений.
- Поддерживают связь за счет отправки назначенному координатором группы брокеру периодических контрольных сигналов.
- Если консюмер прекращает отправку контрольных сигналов, координатор группы признает его неработающим и инициирует перебалансировку.

2. Структура данных

Основные элементы:

- Topics (темы) - сообщения распределяются по топикам (аналог - таблицы);
- Partitions (разделы) - темы разбиваются на партиции (аналог - партиции);
- Batch (пакеты) - набор сообщений, относящихся к одной партиции (сжатие - позволяет передавать и хранить данные более эффективно);
- Message (сообщение) - единица данных (аналог - строка, запись).

Сообщения организованы для хранения в топиках, каждый топик состоит из одной и более партиций, распределённых между брокерами. Распределённость партиций важна для горизонтального масштабирования кластера, так как позволяет клиентам писать и читать сообщения из топиков с нескольких брокеров одновременно, достигая высокой скорости обработки.

Когда новые сообщения попадают в топики кластера, они записываются в одну из партиций этого топика. Сообщения с одинаковыми ключами по правилам записываются в одну партицию, тем самым гарантируя очередность записи и чтения в рамках одной партиции. В случае важности порядка данных, необходимо правильно подобрать бизнес ключ, по которому будут собираться хэш значения.

Гарантия сохранности данных каждой партиции достигается за счет множественной репликации и настраивается фактором репликации (стандарт – 3 копии). Таким образом гарантируется наличие нескольких согласованных копий всех сообщений, хранящихся на разных брокерах и достигается отказоустойчивость кластера в случае сбоев.

Для каждой партиции назначается лидер – брокер, который работает с клиентами и входящими запросами. Лидер партиция всегда принимает сообщения от продюсеров и отдаёт данные консьюмерам, все взаимодействия происходят через него. К лидеру осуществляют запросы его подчиненные – брокеры, которые хранят реплику данных основной партиции.

Перед началом работы, важно определить кто является лидером партиции, поэтому перед записью и чтением клиенты делают запрос метаданных. Причём они могут подключиться к любому брокеру и получить от него нужную информацию.

Метаданные сообщения:

- ключ (запись сообщения в партиции);
- схема (Json/Xml/Avro);
- смещение (Offset).

Распределенный реплицируемый лог является основной структурой данных. Каждая партиция является отдельным реплицируемым логом, хранящимся на диске. Каждое новое сообщение, добавленное продюсером в партицию, сохраняется в начало этого лога и получает уникальный, монотонно возрастающий офсет (число, которое назначается сообщению брокером и отражает порядковый номер сообщения).

Время гарантированного хранения данных на брокере можно контролировать с помощью специальных настроек. Длительность хранения сообщений при этом не влияет на общую производительность системы. Поэтому совершенно нормально хранить сообщения днями, неделями, месяцами или даже годами (в зависимости от задачи).

Заключение

Таким образом, можно выделить следующие различия Apache Kafka с традиционными системами обмена сообщениями:

- Apache Kafka ведет себя как современная распределенная кластерная система способная масштабироваться в пределах достаточных для всех приложений.

- Apache Kafka способна хранить данные столько времени сколько нужно клиентам (дает преимущества при ее использовании в качестве соединительного слоя).
- Apache Kafka позволяет на основе потоков данных вычислять производные потоки и наборы данных динамически (при гораздо меньшем количестве кода).

Литература

1. Шилдт Г. Java 8: руководство для начинающих / Г. Шилдт; пер. с англ. А. Гузикевич. – 6-е изд. – Москва : Вильямс, 2015. – 720 с.
2. Java Concurrency на практике / Г. Брайан [и др.]; пер. с англ. А. Логунов – Санкт-Петербург : Питер, 2019. – 464 с.
3. Hadoop Documentation – URL: https://hadoop.apache.org/docs/stable/ (дата обращения: 15.02.2021).
4. Maven Documentation – URL: https://maven.apache.org/guides/ (дата обращения: 08.03.2021).
5. Уайт Т. Hadoop. Подробное руководство. / Т. Уайт. – Санкт-Петербург : Питер, 2013. – 672 с.
6. Изучаем Spark: молниеносный анализ данных / Х. Карау [и др.]; пер. с англ. – Москва : ДМК Пресс, 2015. – 304 с.
7. Leang B. Improvement of Kafka Streaming Using Partition and Multi-Threading in Big Data Environment. / B. Leang, S. Ean, G.-A. Ryu, et al. // Sensors, 2019. – P. 134–139
8. Shapira G., Narkhede N., Palino T. Kafka: the Definitive Guide. // USA: O'Reilly Media, 2016. – 322 p.
9. Wiatr R., Slota R., Kitowski J. Optimising Kafka for stream processing in latency sensitive systems. // 7 th International Young Scientists Conference in Computational Science, YSC 2018.
10. Hesse G., Lorenz M. Conceptual Survey on Data Stream Processing Systems. // IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), 2015. – P. 797 – 802.
11. Osta M., Bali A., Gherbi A. Event driven and semantic based approach for data processing on IoT gateway devices. // Journal of Ambient Intelligence and Humanized Computing, 2018. – Vol. 5.

УДК *519.688*

ЭКСПЕРИМЕНТАЛЬНОЕ ВЫДЕЛЕНИЕ ОСЦИЛЛОГРАММ, ПОЛУЧЕННЫХ ПРИ ИЗМЕРЕНИИ АРТЕРИАЛЬНОГО ДАВЛЕНИЯ

Д.Г. Усков

Воронежский государственный университет

Введение

Основным показателем состояния пациентов с сердечно-сосудистыми заболеваниями является артериальное давление. Современные приборы наиболее часто используют неинвазивный осциллографический метод. Основой этого метода является нахождение пульсации крупной артерии, в основном плечевой, при ее компрессии или декомпрессии. Пульсации снимаются либо при наборе давления, либо артерия сначала сжимается до уровня давления, превышающее уровень систолического давления, после чего давление постепенно и равномерно снижается. Наиболее распространённые способы получения артериального давления на основе осциллограмм используют технологию нейронных сетей. При обработке искаженных сигналов давления это позволяет повысить точность определения артериального давления у пациентов. Однако такие исследования не позволяют выявлять, анализировать и учитывать факторы, влияющие на результаты измерений.

1. Обработка и анализ данных

Ранее врачами в рамках межцентровых исследований было обследовано около 1000 пациентов и около 4000 измерений. Артериальное давление измерялось несколько раз прибором для получения осциллограмм и после каждого измерения давление контролировалось двумя врачами вручную. Была выделена выборка из более 300 измерений для проведения предварительных исследований. Цель данного исследования определить оказывают ли возрастные изменения и диагноз пациента на результаты измерения артериального давления осциллографическим методом.

Задача данной работы состоит в разработке программного обеспечения для исследования, позволяющего провести оценку влияния исследуемых параметров на амплитуду осциллограмм.

1.1. Входные данные

В качестве входных данных использовались два набора данных (рис. 1). Первый - данные с датчика давления (рис. 1 а), второй – данные с дифференциального датчика (рис. 1 б).

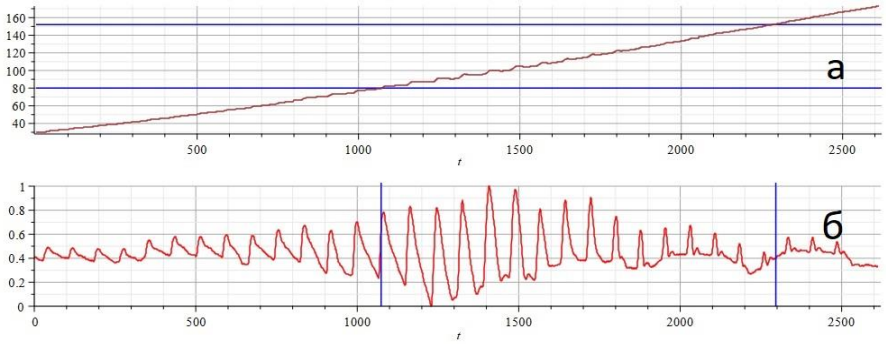


Рис. 1. Пример входных данных.

Необходимо определить амплитуду колебаний кривой с дифференциального датчика в моменты, соответствующие диастолическому и систолическому давлению. Для этого определялись точки пересечения кривой набора давления с уровнем диастолического и систолического давления (150/80 на графике 1 а), затем определялись амплитуды на осциллограмме в эти моменты времени (рис. 1 б). Данные с дифференциального датчика были нормированы.

1.2. Нормированные амплитуды систолического и диастолического давления

Нахождение нормированных амплитуд производилось в три шага — выделение данных для обработки, построение огибающих и непосредственное вычисление коэффициентов.

Из массива данных выделяются записи, попадающие в определенный врачами диапазон систолического и диастолического давления. Для этого выбираются позиции I_u - первая строка, давление на которой равно или меньше определенному врачем систолическому давлению, и I_l - первая строка, на которой уровень давления, определенного внешним датчиком, больше или равен диастолическому. После чего диапазон таких данных расширяется на 20% для упрощения построения огибающих. Пример данных до выделения показан на рисунке 2, где вертикальными линиями отмечен необходимый диапазон:

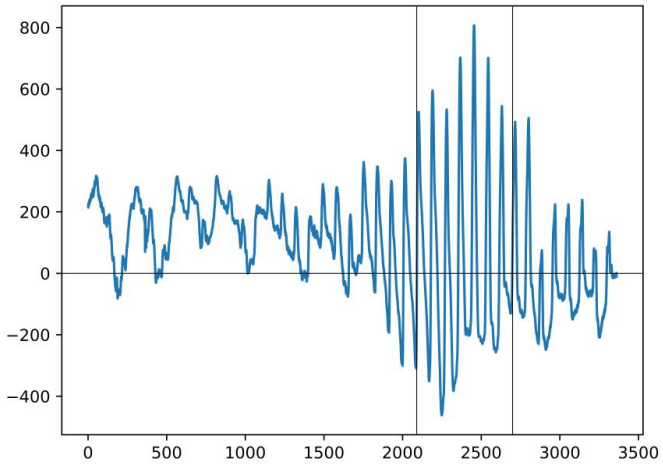


Рис. 2. Пример данных, полученных с внутреннего датчика.

Далее, для кривой, образованной данными дифференциального датчика строятся нижняя и верхняя огибающие через определение локальных минимумов. Это разбиение проводится для улучшения результатов построения огибающей. Пример выделения данных и построения огибающих предоставлен на рисунке 3:

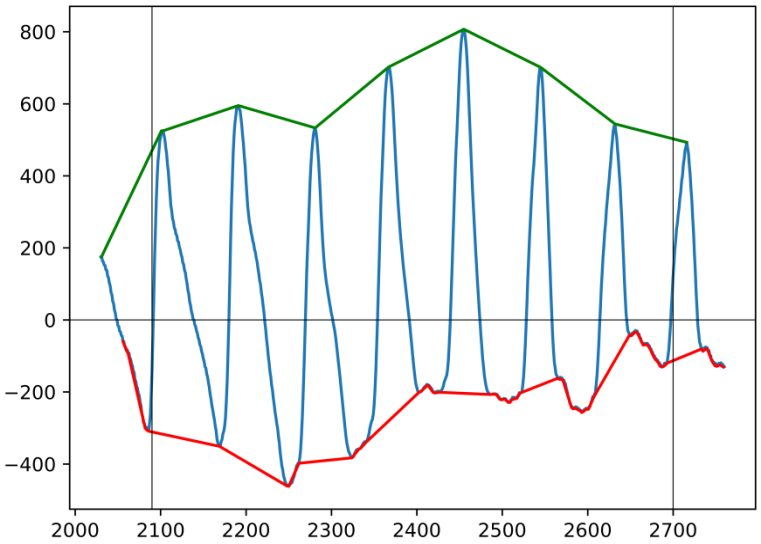


Рис. 3. Пример построения огибающих.

После построения производных вычисляется сумма их модулей их массивов $\mathbf{d} = |\mathbf{d}_u| + |\mathbf{d}_l|$, где $\mathbf{d}_u$ - массив данных нижней огибающей, а $\mathbf{d}_u$ - массив данных нижней огибающей. После этого массив сумм нормализуется путем деления каждого из его элементов на максимальный элемент этого массива. Для нахождения систолического и диастолического нормированных амплитуд выбираются элементы массива, соответствующие строкам, на которых были достигнуты уровни систолического и диастолического давления соответственно.

2. Анализ результатов

После нахождения нормированных амплитуд были построены гистограммы для пациентов по выбранным критериям. Для всего набора данных гистограммы для диастолического (рис.4) и систолического (рис. 5) имеют следующий вид:

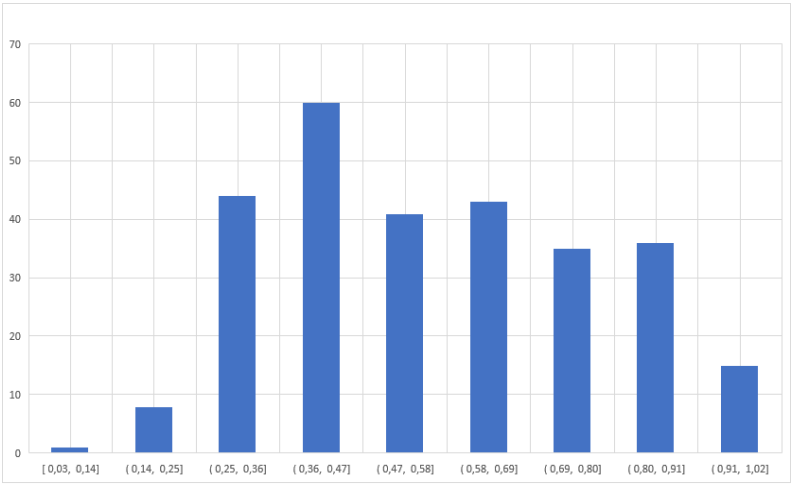


Рис. 4. Гистограмма диастолического давления.

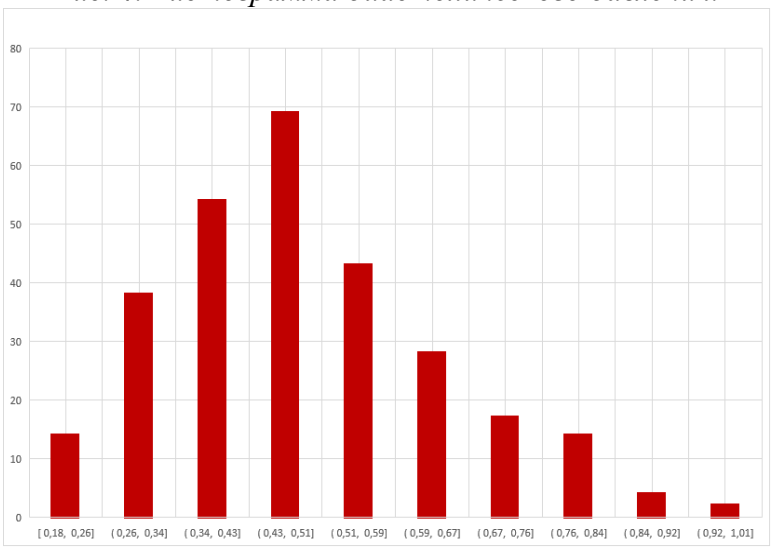


Рис. 5. Гистограмма систолического давления.

Диаграммы показывают, что для диастолического и систолического давлений нормированные амплитуды имеют различные значения. Анализ гистограмм для здоровых и пациентов с заболеваниями различаются не значительно. Для анализа собранных данных была построена гистограмма по возрастным группам, которая показана на рисунке 6:

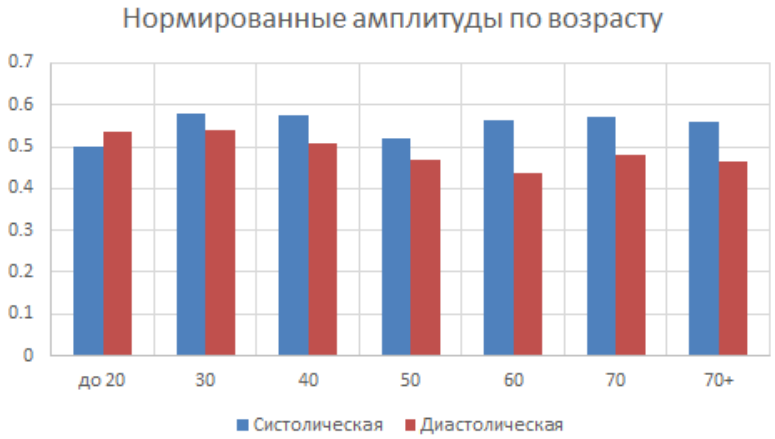


Рис. 6. Гистограмма систолического давления.

Как видно из рисунка, для людей до 20 лет, нормированная амплитуда систолического давления превышает амплитуду диастолического, однако для людей старше 20 наоборот, диастолическая амплитуда превышает систолическую. Нужно отметить, что в период от 20 до 30 лет наблюдается рост обоих амплитуд, что совпадает с повышением роста уровней систолического и диастолического давления в процессе роста человека.

Кроме того, с периода от 30 до 60 лет нормированная амплитуда систолического давления резко падает, достигая своего минимального значения в 60 лет и стабилизируясь в возрасте 70+ лет. Ее рост в возрасте от 60 до 70 лет может быть вызван нехваткой данных для данного возрастного промежутка, что привело к незначительному повышению. Нормированная амплитуда диастолического давления также понижается в период от 30 до 50 лет, однако он возвращается к своему уровню до снижения значению в 60 лет и остается в нем до конца жизни. В целом, тенденция снижения коэффициентов связана с постепенным ростом уровня артериального давления по мере старения пациента.

Заключение

Проведенный анализ выявил определенные тенденции в найденных параметрах, которые в целом совпадают с общеизвестными изменениями уровней артериального давления по мере взросления пациентов. В дальнейшем планируется проведение повторного анализа с использованием альтернативных методов построения огибающей и применением высокочастотных фильтров для получения более наглядного результата.

Литература

1. Скоробогатова А. И. Метод измерения артериального давления с использованием нейронных сетей / А. И. Скоробогатова, А. А. Анисимов // Известия СПбГЭТУ «ЛЭТИ» № 10/2017, с. 75 - 84.

2. Лебедев В.И. Функциональный анализ и вычислительная математика / В.И. Лебедев; Москва: Физматлит, 2000 г. – 292 с.

3. Петин В.Г. Проекты с использованием микроконтроллера Arduino / Виктор Петин; Санкт-Петербург: БВХ-Петербург, 2015. – 401 с

4. Абакунов Н.В. Методика математического моделирования сердечно-сосудистой системы / Н.В. Абакунов // Математическое моделирование: сборник научных трудов. – 2000. – №3 – С. 106 – 117.

5. Andres James, inventor; Siemens Medical Electronics Inc., assignee. Pulse signal extraction apparatus for an automatic blood pressure gauge. United States patent US 5,355,890 1998 October 18.

УДК 004.6 : 004.91

ПРОБЛЕМЫ ФОРМИРОВАНИЯ И КОНВЕРТАЦИИ ДОКУМЕНТОВ НА ПРИМЕРЕ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЯЮЩЕЙ КОМПАНИИ

А. О. Федорин, К. Г. Резников

Воронежский государственный университет

Введение

Управляющие компании имеют дело с множеством задач, таких как управление имуществом, взаимодействие с жильцами, координация работы сотрудников и прочие. Современные интернет-технологии и технологии разработки программного обеспечения могут помочь упростить и автоматизировать многие из этих задач, что позволит компаниям стать более эффективными и предоставлять более качественные услуги своим клиентам. Использование веб-приложений, как для управляющих компаний, так и в общем имеет ряд преимуществ.

Во-первых, веб-приложение может быть доступно на любом современном вычислительном устройстве с предустановленным веб-браузером и подключением к Интернет-сети. Это позволяет управляющим компаниям не зависеть от технического оснащения сотрудников и клиентов, снижает минимальные системные требования и ограничения вычислительных систем, что делает их использование очень удобным. Пользователи не зависят от используемой операционной системы и типа устройства [1].

Во-вторых, веб-приложения могут быть максимально эффективно настроены для автоматизации рутинных задач, что позволяет увеличить эффективность работы компании, снизить нагрузку на сотрудников, быстрый доступ клиентов к личному учету без посещения центров обслуживания.

В-третьих, это дает компании быстрый и полный доступ ко всей информации ее работы, такой как отчеты, квитанции, статистику и учетную информацию.

Особенно важной функцией веб-приложений для управляющих компаний является функционал управления запросами жильцов и обработки заявок, что помогает улучшить обслуживание клиентов, снизить или полностью исключить очереди и посещения офисов, избегать ошибок за счет цифровых технологий, что в свою очередь значительно повышает качество обслуживания.

Для реализации такого веб-приложения существует множество современных технологий.

Обычно для реализации бизнес-логики приложения используют фреймворки. Например, такие как Django, Ruby on Rails, Express.js и другие [2]. Они определяют архитектуру приложения, язык программирования, на котором будет описана логика, а также могут включать в себя готовые модули, которые реализуют некоторый функционал.

Для создания пользовательского интерфейса используют язык разметки HTML и каскадные таблицы стилей CSS, а также язык программирования JavaScript, который интерпретируется веб-браузером для интерактивного взаимодействия пользователя с веб-приложением [3].

Для хранения и управления данными используют базы данных, самыми распространёнными из которых являются MySQL, PostgreSQL и MongoDB.

Во время работы управляющей компании, есть необходимость работать с большим количеством документов, преимущественно в формате PDF, так как он является наиболее легким и удобным для отображения, а также поддерживаемого всеми современными печатными системами. Используя веб-приложение, данные, представленные в документе можно хранить в базе данных приложения, но это не исключает необходимость иметь возможность выгружать их в формате PDF. Для предоставления подобного функционала удобнее всего использовать веб-шаблоны [4] документа, данные которого хранятся в базе данных и которые, при необходимости, формируют документ в требуемом формате. При такой архитектуре важно озаботиться проблемой правильной конвертации подобного документа из веб-шаблона, а также проблемой производительности приложения при таком функционале, что будет раскрыто во второй главе. Также рассмотрим преимущества и недостатки, связанные с использованием веб-приложений при решении поставленной задачи.

Важно рассмотреть, как современные технологии для веб-разработки позволяют решить проблемы формирования и конвертации документов на примере работы управляющей компании, чтобы улучшить качество предоставляемых услуг.

1. Инструменты и технологии разработки

В демонстрируемой реализации для описания логики использовался фреймворк Django [5] на языке Python. В качестве базы данных была выбрана PostgreSQL [6]. Фреймворк Django позволяет увеличить скорость разработки, благодаря готовым и хорошо проработанным модулям, например, как URL модули и модули по обработке ошибок. Также фреймворк обеспечивает высокий уровень безопасности за счет встроенных механизмов аутентификации и авторизации. База данных же была выбрана по принципу лучшей совместимости с данным фреймворком. Для связи Django и PostgreSQL используется модуль ORM (англ. Object Relational Mapping).

В работе приложения присутствуют задачи, для которых необходимо использовать дополнительные инструменты, так как веб-приложения имеют ряд ограничений. Использование дополнительных библиотек и технологий позволит расширить возможности и улучшить скорость работы приложения.

Для управления задачами приложения и фоновое их выполнение потребуется модуль Celery [2] для асинхронной очереди задач. Celery позволяет вынести высоконагруженные задачи из основного приложения и выполнять их асинхронно в фоновом режиме. Например, такие задачи будут использоваться для массовой рассылки счетов оплаты.

Для ускорения выполнения конвертации документов воспользуемся модулем WebAssembly [7]. Данная библиотека позволяет воспроизводить алгоритм на других языках программирования, а также выполнять его на стороне клиента, что позволит снизить нагрузку на сервер. Также для конвертации понадобится использовать язык программирования библиотеки, компилируя исходный код в WebAssembly.

Для этого был выбран язык Rust и дополнительная библиотека Headless Chrome, которая позволит сохранять изображение веб-страницы по ссылке.

Воспользуемся библиотекой Headless Chrome на языке Rust для создания стандартной функции преобразования веб-страницы в файл формата PDF [8] и компиляции ее в код WebAssembly. Используем библиотеку *wasm-bindgen* для генерации JavaScript API и экспорта Rust функций [9].

Библиотека *wasm-bindgen* используется для связывания кода, написанного на Rust, с JavaScript или TypeScript [10], который работает на WebAssembly. Она упрощает взаимодействие между Rust и JavaScript, позволяя передавать объекты между двумя языками. С помощью этой библиотеки будут вызываться функции Rust из JavaScript и наоборот.

При этом с помощью этой библиотеки автоматически создается JavaScript-интерфейс на основе WebAssembly-модуля. Это упрощает работу с WebAssembly в Django проекте и позволяет быстрее интегрировать его в приложение.

Далее полученный WebAssembly модуль добавляется в Django проект и используется для конвертации.

Таким образом получена система хранения документов с минимальными требованиями к производительности сервера и хорошей скоростью работы для конечного пользователя. При этом у сотрудников всегда есть доступ к документам в виде веб-страницы и быстрый способ получить их PDF версию, без нагрузки на сервер.

Однако, использование WebAssembly в Django приложении также может иметь некоторые недостатки, например, большой размер бинарных файлов, сложность отладки и тестирования кода, а также недостаточную поддержку некоторыми инструментами и фреймворками. В данном случае это несущественные недостатки, так логика под WebAssembly достаточна проста и не будет занимать много места на сервере. А фреймворки, используемые в проекте отлично поддерживают данную технологию.

Рассмотрим более простое решение для конвертации файлов в формат PDF, используя Python библиотеку WeasyPrint [11] на стороне сервера [12].

WeasyPrint является библиотекой для создания документов, которые требуют точного воспроизведения веб-страниц. Она использует современные веб-технологии, такие как браузерный движок, чтобы создавать документы высокого качества, с учетом содержания сложных макетов, графики и шрифтов. Данная библиотека наиболее точно подходит для создания файлов PDF формата на основе веб-страниц.

2. Программная реализация

Базовая архитектура приложения основана на архитектуре, представляемой фреймворком Django.

Django позволяет разрабатывать веб-приложения, используя шаблон MVC (Model-View-Controller с англ. Модель-Представление-Контроллер). Модель представляет собой данные приложения, представление отвечает за отображение этих данных пользователю, а контроллер управляет взаимодействием между моделью и представлением. Выделим некоторые особенности данного шаблона.

В компоненте Модель (Model) для работы с базами данных Django использует ORM, что позволяет работать с данными в виде объектов Python, не используя SQL-запросы напрямую. Тем не менее ими можно воспользоваться при необходимости при более сложном использовании.

В компоненте представление (View) используем систему шаблонизации, которая позволяет отделить представление от логики приложения, облегчая поддержку и изменение пользовательского интерфейса.

В компоненте контроллер (Controller) Django имеет сложную систему URL-адресации, которая позволяет определять маршруты для запросов, поступающих на сервер, а после направлять их на соответствующие контроллеры.

В проекте общая логика, описание таблиц базы данных, а также контроллеры распределяются по приложениям внутри структуры проекта. Таким образом, вся сложная логика приложения разделена по небольшим дочерним приложениям, отвечающим за свои функции. Приложение для управляющей компании имеет следующий необходимый функционал, разделенный на дочерние приложения:

- управление домами (отвечает за хранение, предоставление и обработку информации о домах, квартирах и лицевых счетах)

- управление платежами (отвечает за формирование начислений и загрузку платежей со сторонних сервисов)
- управление сотрудниками (отвечает за аутентификацию сотрудников)
- управление документами (отвечает за хранение шаблонов документов и конвертацию в необходимые форматы.

Рассмотрим подробно структуру базы данных.

База данных веб-приложения для управляющей компании должна быть структурирована таким образом, чтобы хранить всю необходимую информацию для эффективного управления компанией.

Перечислим необходимые сущности для формирования таблиц базы данных приложения.

- управляющая компания (юридическая информация) – *Company*,
- дом – *House*,
- помещение (квартира) – *Room*,
- собственник – *Owner*,
- лицевой счет – *PersonalAccount*,
- период (необходимы для разделения промежутков времени, за которые начисляется счет для оплаты) – *Period*,
- услуга (информация об услуге, за которые компания выставляет счет) – *Service*
- тариф (информация о цене единицы потребляемой услуги) – *ServiceTariff*,
- начисление (информация о выставленном счете за определенный период) – *ServiceValue*,
- прибор учета (информация о приборе учета, дате установке, проверке, а также квартиры, в которой установлен) – *Device*,
- показание прибора учета – *DeviceValue*,
- платежей – *Payment*,
- пользователь (данные о сотруднике управляющей компании) – *User*,
- документ – *Document*.

Схема база данных представлена на рисунке 1.

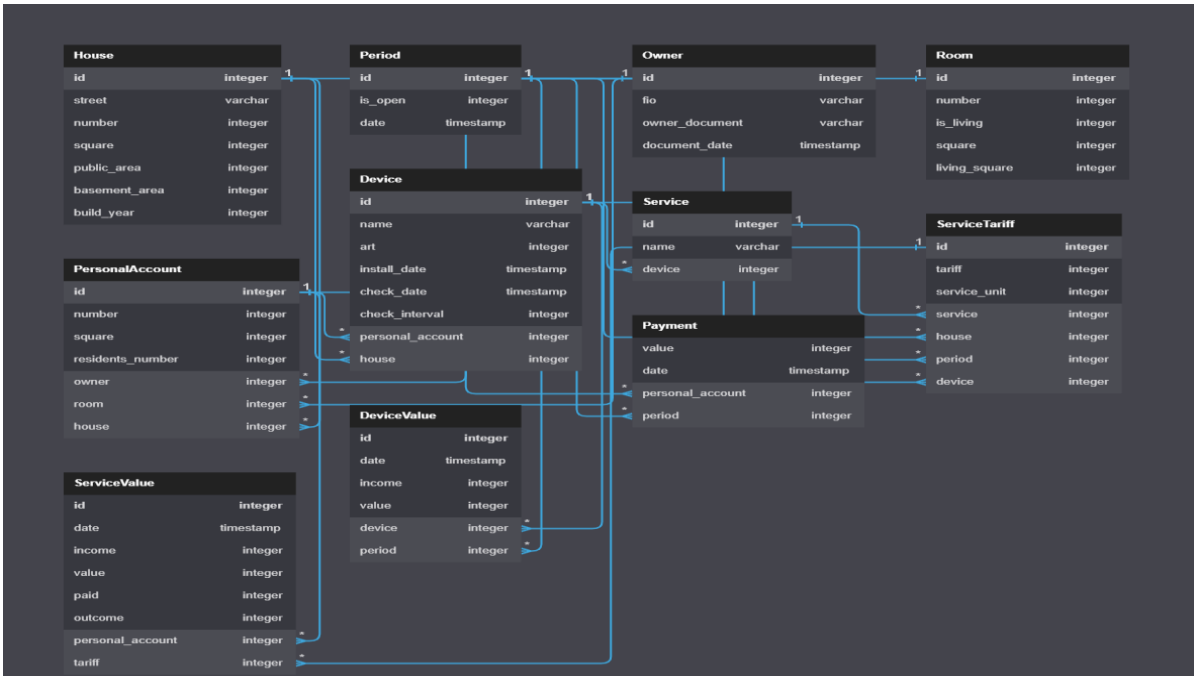


Рис. 1. Схема базы данных

Представленная структура база данных является необходимой для минимальной работы учета управляющей компании. За счет преимущества веб-приложения, такая база данных может быть легко масштабируема под нужды компании. Также она может использоваться и в других приложениях управляющей компании посредством API. Например, для такой реализации можно использовать Django Rest Framework [13].

После создания базовой архитектуры приложения необходимо реализовать отдельные модули, которые будут выполнять необходимые задачи, а именно конвертации документа из веб-формата в PDF.

Для начала опишем, каким образом хранятся документы и каким образом к ним должен быть предоставлен доступ.

По умолчанию доступ к документам в формате веб-страницы осуществлен из административной панели, к которой имеют доступ сотрудники компании, но для реализации функционала с использованием WebAssembly необходимо предоставить доступ к странице документа и программе WebAssembly, так как программа работает в виртуальной среде и имеет ограниченные возможности доступа к файловой системе и сети.

Для этого к каждому документу создаются дополнительные уникальные HASH-ключи, по котором возможно обратиться к страничке данного файла. HASH-ключ никаким образом не будет использоваться в обычной работе программы, так что возможность свободного доступа к счетам жильцов компании минимальна.

Таким образом получим безопасный доступ к документам для программы на WebAssembly.

3. Анализ использования методов и результаты конвертации

Рассмотрим быстродействие двух методов конвертации веб-страницы, используя библиотеки WeasyPrint для Python и HeadlessChrome для WebAssembly. В таблице 1 представлены результаты работы двух скриптов при разном количестве генерируемых файлов. Результаты контрольных измерений представлены в секундах.

Таблица 1

Сравнение методов конвертации

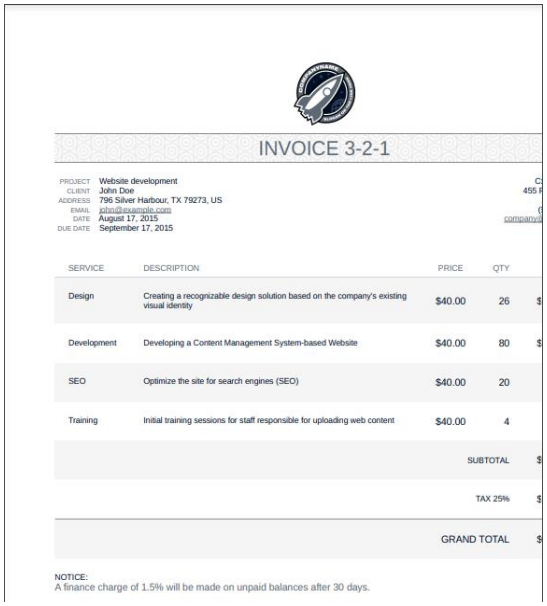
Кол-во	**WeasyPrint**	**Rust**
1	1	3
10	8	27
50	47	172

На основе полученной таблицы имеем следующий вывод. Библиотека WeasyPrint имеет преимущество в скорости работы в сравнении с библиотекой Headless Chrome для языка Rust. Это объясняется более ресурсоемким браузерным движком, который используется в библиотеке Headless Chrome. Тем не менее при увеличении количества генераций и тот и другой метод требуют значительное время на выполнение конвертации, что в случае работы на стороне сервера имеет больший негативный эффект на общую работу веб-приложения, чем незначительно большее время работы одной задачи на стороне клиента.

Также рассмотрим различия в генерируемых PDF файлах на примере стандартного HTML шаблона счета.

На рисунке 2, представлен результат работы библиотеки WeasyPrint. Изображение обрезано по правой стороне, в то время как библиотека Headless Chrome результат которой представлен на рисунке 3, захватывает всю ширину страницы. Это не является критической разницей при использовании веб-шаблонов, но с точки зрения удобства использования и требуемой оптимизации в данной задаче библиотека Headless Chrome является более простой в

использовании.



INVOICE 3-2-1

PROJECT Website development
CLIENT John Doe
ADDRESS 796 Silver Harbour, TX 79273, US
EMAIL john@example.com
DATE August 17, 2015
DUE DATE September 17, 2015

SERVICE	DESCRIPTION	PRICE	QTY	
Design	Creating a recognizable design solution based on the company's existing visual identity	$40.00	26	$
Development	Developing a Content Management System-based Website	$40.00	80	$
SEO	Optimize the site for search engines (SEO)	$40.00	20	
Training	Initial training sessions for staff responsible for uploading web content	$40.00	4	
			SUBTOTAL	$
			TAX 25%	$
			GRAND TOTAL	$

NOTICE:
A finance charge of 1.5% will be made on unpaid balances after 30 days.

Рис. 2. Результат конвертации WeasyPrint

Однако, если оптимизировать процесс конвертации WeasyPrint и свести к минимуму сложно вычислимые свойства стилизации, которые рекомендованы в официально документации библиотеки [14], то возможно получить минимально возможное время конвертации файлов и точной отрисовки документов.

INVOICE 3-2-1

PROJECT Website development
CLIENT John Doe
ADDRESS 796 Silver Harbour, TX 79273, US
EMAIL john@example.com
DATE August 17, 2015
DUE DATE September 17, 2015

Company Name
455 Foggy Heights,
AZ 85004, US
(602) 519-0450
company@example.com

SERVICE	DESCRIPTION	PRICE	QTY	TOTAL
Design	Creating a recognizable design solution based on the company's existing visual identity	$40.00	26	$1,040.00
Development	Developing a Content Management System-based Website	$40.00	80	$3,200.00
SEO	Optimize the site for search engines (SEO)	$40.00	20	$800.00
Training	Initial training sessions for staff responsible for uploading web content	$40.00	4	$160.00
			SUBTOTAL	$5,200.00
			TAX 25%	$1,300.00
			GRAND TOTAL	$6,500.00

NOTICE:
A finance charge of 1.5% will be made on unpaid balances after 30 days.

Рис. 3. Результат конвертации Rust

4. Оптимизация шаблонов для конвертации в PDF файлы

Библиотека WeasyPrint настроена таким образом, чтобы максимально быстро создавать документы в формате PDF и исключать сложные свойства стилизации, которые усложняют работу интерпретатора веб-браузера.

Важно исключить свойства, относящиеся к браузерным спецификациям определенных

версий, например, *–webkit* для браузеров на движке Chromium или *–ms* для браузера Mozila Firefox. Вместо таких свойств необходимо использовать принятые в стандарте свойства. Например, для свойств Flex контейнеров использовать только установленные стандартом *flex*. Для таких свойств есть ряд браузерных альтернатив: *-webkit-box, -ms-flex* и т.д., которые необходимо исключить. Множество проектов настроены таким образом, что данные свойства при сборке проекта добавляются в итоговый файл для оптимизации веб-приложений под большинство браузеров. Но в случае с работой для конвертации шаблонов в файлы PDF формата, необходимо переконфигурировать сборку так, чтобы отключить добавление таких свойств.

Вторым важным фактором оптимизации является настройка страницы для генерации файлов в формате PDF. Необходимо корректно настроить *@page* свойства, которые отвечают за печать веб-страниц. Набор данных свойств отвечает за отступы боковых интервалов, колонтитулы, нумерация страниц и прочие параметры, необходимые при создании файлов PDF формата. При отсутствии настроек таких параметров, параметры устанавливаются по умолчанию, а в следствии чего документ получается не таким, каким его ожидает увидеть пользователь и не таким, каким он выглядит при просмотре HTML страницы в веб-браузере. Результат без настройки параметров представлен на рисунке 2.

Рассмотрим пример оптимизации HTML файла для создания билетов в формате PDF.

Структура билета выглядит как сетка разметки времени, ряда и места. Как правило для сеток используют свойство *grid*, но данной свойство не поддерживает библиотека WeasyPrint. В таком случае необходимо использовать альтернативу. С помощью тегов *dd* и *dt*, которые предназначены для описания терминов. Опишем стилизацию таким образом, что наименование будет являться заголовком сетки, а описание значением в сетке. Через описание стилей для тега *dl* опишем набор колонок. В текущем случае нельзя использовать в качестве сетки HTML таблицу, так как таблица не поддерживает глубокой стилизации.

Задав стили и настройку страницы через свойство *@page*, получим следующий результат, представленный на рисунке 4.

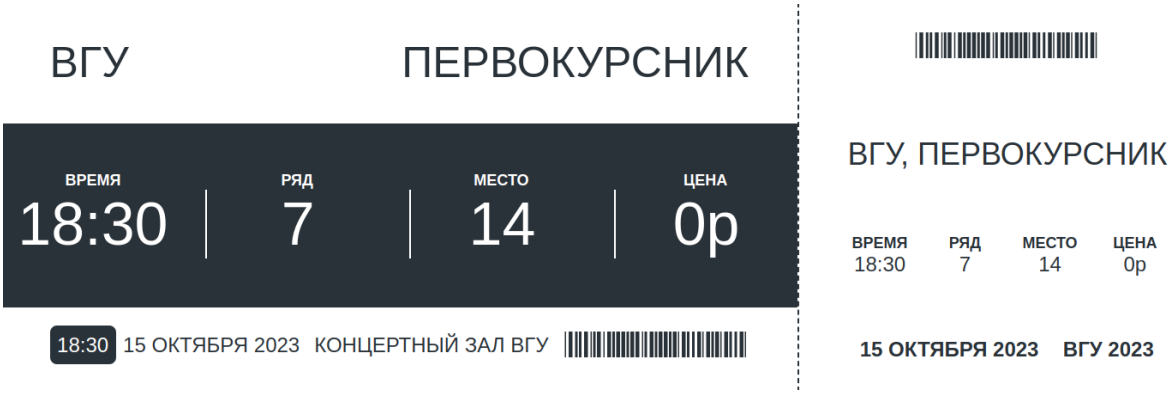


Рис. 4. Изображение PDF файла

Заключение

В данной статье рассмотрен пример разработки приложения для управляющей компании с целью повышения качества предоставления услуг, повышения производительности, снижению нагрузки на сотрудников, уменьшение очередей в филиалах управляющих компаний, более удобного использования, передачи и хранения информации.

Проблема управляющих компаний является актуальной в настоящее время. Способы решения ряда проблем в документо-обороте управляющих компаний описаны в настоящей

статье.

Подробно описаны технологии, которые используются в разработке приложения. Описаны преимущества представленных подходов к разработке. Описана структура разработанного приложения, сопутствующие технологии и способы их использования. Продемонстрирована разработанная структура базы данных и итоговая схема связей сущностей. Показан способ использования конвертации файлов через приложение, без необходимости хранения итоговых документов на сервере. Продемонстрированы примеры конвертации файлов различными способами. Реализован и представлен способ организации очереди конвертации ряда файлов. Проведен анализ, выявлены достоинства и недостатки способов конвертации. Описаны подходы для оптимизации конвертации файлов. Реализован пример с оптимизаций для конвертации HTML шаблона в документ формата PDF.

Список литературы

1. Резников К. Г. Разработка программного обеспечения для визуализации трехмерных поверхностей в веб-браузере / К. Г. Резников, С. Н. Медведев // Вестник ВГТУ. Том 17, №6. – Воронеж: ВГТУ, 2021 - С. 13-19.
2. Дронов В. А. Django 3.0. Практика создания веб-сайтов на Python / В. А. Дронов. – СПб. : БХВ-Петербург, 2021. – 704 с.
3. Фримен, Э. Изучаем программирование на JavaScript / Э. Фримен, Э. Робсон. – СПб. : Питер, 2015. – 640 с.
4. Веб-шаблоны в Django. – Режим доступа: https://docs.djangoproject.com/en/4.1/topics/templates/. – (Дата обращения: 15.04.2023).
5. Грофф, Д. Р. SQL: полное руководство / Д. Р. Грофф, П. Н. Вайнберг, Э. Д. Оппелъ. – 3-е изд. – М. : ООО "И.Д. Вильяме", 2015. – 960 с.
6. Документация Django. – Режим доступа: https://django.fun/ru/docs/django/4.1. – (Дата обращения: 15.04.2023).
7. Шлиттен Б. WebAssembly – Полное руководство: Безопасный, быстрый и портативный код / Б. Шлиттен. – O'Рейли Медиа : Клифорния, 2022. – 400 с.
8. Язык программирования Rust. – Режим до-ступа: https://rust-lang.ru. – (Дата обращения: 15.04.2023).
9. Галлан, Ж. Ж. WebAssembly в действии / Ж. Ж. Галлан. – СПб. : Питер, 2022. – 496 с.
10. Черный Б. Профессиональный TypeScript. Разработка масштабируемых JavaScript приложений / Б. Черный. – СПб. : Питер, 2022. – 352 с.
11. WeasyPrint. The Awesome Document Fatory – Режим доступа: https://weasyprint.org. – (Дата обращения: 15.04.2023).
12. Уилкс М. Профессиональная разработка на Python / М. Уилкс. – ДМК Пресс, 2021. – 502 с.
13. Документация Django Rest Framework. – Ре-жим доступа: https://www.django-rest-framework.org/. – (Дата обращения: 15.04.2023).
14. WeasyPrint. 59.02B documentation – Режим доступа: https://doc.courtbouillon.org/weasyprint. – (Дата обращения: 16.04.2023).

УДК 004.94

ОБЗОР КОМПОНЕНТОВ ПЛАГИНА GAMEPLAY ABILITY SYSTEM ДЛЯ UNREAL ENGINE.

А.Е. Холодова. Е.В. Трофименко

Воронежский государственный университет

Введение

Gameplay Ability System (GAS) – разработанный командой Epic Games плагин, позволяющий реализовать довольно гибкую систему способностей и эффектов. Этот плагин уже используется в индустрии в играх Fortnite и Paragon. К достоинствам GAS можно отнести ее гибкость, легкость в подключении; учтен Multiplayer режим и его особенности, а также что система может использоваться после ее инициализации без участия программиста.

1. Основные компоненты плагина GAS.

1.1. Компонент Ability System Component (ASC)

Ability System Component (ASC) – главный компонент и основа всей системы. Он ответственен за обработку всех эффектов и способностей персонажей (удаление, добавление), обновление атрибутов в сетах атрибутов при применении эффектов и способностей. Использовать все возможности GAS смогут только Actor, имеющие этот компонент.

Для полноценной работы с этим компонентом рекомендуется реализовать интерфейс IAbilitySystemInterface, в котором необходимо переопределить геттер GetAbilitySystemComponent() для ASC. ASC компоненты «общаются» друг с другом через этот интерфейс.

Активные на данный момент эффекты ASC содержит в ActiveGameplayEffects типа FActiveGameplayEffectsContainer. Активные способности – в ActivatableAbilities типа FGameplayAbilitySpecContainer. Если необходима итерация по ActivatableAbilities.Items, то перед циклом необходимо добавить ABILITYLIST_SCOPE_LOCK(), чтобы предотвратить изменение этого списка.

Существует 3 режима репликации для GameplayEffect, GameplayTag, and GameplayCue:

1. Полная репликация. Лучше использовать в Single Player играх. Каждый эффект копируется на все клиенты.
2. Смешанная репликация. Обычно используется в Multiplayer играх для акторов, которых контролирует игрок. Эффекты копируются на клиенте владельца. Только теги и GameplayCue копируются на все клиенты.
3. Минимальная репликация. Multiplayer игры для ИИ Акторов. Эффекты не копируются никогда. Только теги и GameplayCue копируются на все клиенты.

ASC компонент также может находиться на PlayerState. Обычно такое решение подходит, если состояние Actor, его атрибуты и эффекты (GameplayEffect) необходимо сохранять после его респауна (появления в заданной локации). При такой реализации необходимо увеличить NetUpdateFrequency, иначе могут появиться задержки и лаги при работе GAS.

Смешанная репликация ожидает, что OwnerActor ASC является Controller. По умолчанию, владелец PlayerState является Controller, а Character нет, и поэтому необходимо вызвать SetOwner() на OwnerActor с валидным контроллером для корректной работы.

В класе AbilitySystemGlobals хранится глобальная информация о GAS, и большинство переменных этого класса могут быть установлены в файле DefaultGame.ini.

1.2. Компоненты Attribute и Attribute Set

Атрибуты (Attribute) – это float значения, определенные в структуре FGameplayAttributeData, содержащей текущее (CurrentValue) и базовое (BaseValue) значения. Базовое значение неизменно, а текущее состоит из базового значения, к которому прибавлены модификаторы эффектов (GameplayEffect). Атрибуты обычно используются для обозначения всех численных значений, относящихся к Actor и которые могут быть изменены эффектами и способностями, например, здоровье, уровень героя, количество зарядов у зелья или заклинания.

Максимальные значения для атрибутов обычно хранятся в других отдельных атрибутах. Но также возможно определить DataTable типа FAttributeMetaData, в которой указаны максимальные и минимальные значения. Однако Epic Games все еще отмечают данную структуру как «работу в процессе».

Атрибуты могут быть определены только в C++ в файле AttributeSet.h. Также рекомендуется добавить макрос ATTRIBUTE_ACCESSORS, автоматически генерирующий геттеры и сеттеры для атрибутов.

Инициализация атрибутов может быть проведена с использованием макроса, но данная практика не очень рекомендуется. Epic Games советуют использовать мгновенный (instant) эффект.

AttributeSet – контейнер, в котором хранятся атрибуты, отвечающий за репликацию и управление изменениями над ними. Ожидается, что разработчики создадут свой производный класс от UAttributeSet.

ASC может иметь как один, так и несколько AttributeSet, так как они незначительно нагружают память. Например, игровому персонажу маг может понадобиться набор с магическими атрибутами, а войну – набор с атрибутами выносливости и силы. Но нельзя использовать более чем один набор одного класса на одном ASC; один из таких AttributeSet будет выбран, остальные - проигнорированы.

Meta Attribute – атрибут, хранящий временное значение, например, урон. Вместо прямого изменения атрибута здоровья эффектом часто используется Meta Attribute урона. Таким образом можно модифицировать это значение в GameplayEffectExecutionCalculation или самим AttributeSet. Например, вычесть урон сначала из атрибута щита, а затем оставшееся из атрибута здоровья. Обычно Meta Attribute не подлежат репликации.

Для прослушивания изменений атрибута используется функция GetGameplayAttributeValueChangeDelegate(FGameplayAttribute Attribute), находящаяся в UAbilitySystemComponent. Эта функция возвращает делегат, на который можно подписаться. Делегат имеет параметры новое и старое значения и FGameplayEffectModCallbackData, который устанавливается только на сервере.

Для создания производного атрибута, зависящего от одного или более других атрибутов, используется бесконечный эффект с одним или более основанных на атрибуте или MMC модификаторы. Производный атрибут обновляет свое значение автоматически вместе с базовыми атрибутами.

AttributeSet можно добавлять и убирать с ASC в реальном времени, но удаление может быть опасным. Если на клиенте удалить AttributeSet, а с сервера будет реплицировано значение его атрибутов, то атрибут не найдет свой AttributeSet и произойдет краш.

Функции PreAttributeChange(const FGameplayAttribute& Attribute, float& NewValue) и PostGameplayEffectExecute(const FGameplayEffectModCallbackData & Data) вызываются до и после обновления значения атрибута. Первая обычно используется для clamping значений (но не для вызова геймплей событий), вторая – для манипуляций над атрибутами (например, можно вычесть только что вычисленное значение атрибута урона из атрибута щита или воспроизвести анимацию и т.п.).

1.3. Компонент Gameplay Tag

Теги (FGameplayTags) – иерархические имена вида ТегРодитель.ТегРебенок.ЕщеТег… При добавлении эффекта огненного урона от заклинания можно добавить на Actor тег Магия.Урон.Огонь. Тегами можно заменить то, что раньше выполнялось при помощи булевых значений или перечислений.

Теги должны быть определены заранее в файле DefaultGameplayTags.ini. В редакторе Unreal Engine существует интерфейс работы с тегами. Теги обычно добавляются на ASC, который реализует интерфейс IGameplayTagAssetInterface с функциями, дающими доступ к тегам компонента.

Несколько тегов можно хранить в контейнере тегов (FGameplayTagContainer), который предпочтительнее использовать чем простой массив тегов, так как методы контейнера оптимизированы. ASC предоставляет делегат, вызываемый при добавлении или удалении тегов, на который можно подписаться. В GameplayTagManager определены функции для более углубленной работы с тегами.

1.4. Компонент Gameplay Effect

Эффекты (GameplayEffects) – средства, через которые способности изменяют атрибуты и теги на своем ASC и на других. Класс UGameplayEffect предназначен быть классом только для данных, определяющий один эффект, и никакой другой логики не нужно добавлять.

Эффекты изменяют атрибуты через модификаторы и GameplayEffectExecutionCalculation, могут добавлять или запускать GameplayCue или выдавать новые способности для ASC (только длительные и бесконечные эффекты), могут давать неуязвимость и другое на основе тегов.

Есть три типа продолжительности эффектов: мгновенные, длительные и бесконечные. Мгновенные перманентно изменяют базовое значение атрибута, никакие теги не применяются. Длительные временно изменяют текущее значение атрибута и применяют теги, которые снимутся после удаления эффекта, их длительность задается в классе эффекта. Бесконечные действуют как длительные эффекты, но не могут сами закончиться, убираются только вручную через ASC или способность.

Длительные и бесконечные эффекты могут накладывать периодичные эффекты, которые применяют модификаторы и GameplayEffectExecutionCalculation через указанный период. Они действуют как мгновенные эффекты и подходят для длительного урона (damage over time). Длительные и бесконечные эффекты могут временно отключаться и включаться после применения, если не соблюдаются критерии для их выполнения. Но при отключении эффект не удаляется, только выключаются модификаторы и теги.

Обычно экземпляры эффектов не создаются. Когда способностям или ASC нужно применить эффект, они создают GameplayEffectSpec из дефолтного объекта класса эффекта. Далее успешно примененные GameplayEffectSpec добавляются в новую структуру FActiveGameplayEffect, которая хранится ASC в специальном контейнере ActiveGameplayEffects. GameplayEffectSpec можно назвать экземплярами эффектов. Они могут быть легко созданы и изменены в runtime, а эффекты обычно создаются дизайнерами заранее, так как создавать их в runtime сложно. GameplayEffectSpec обычно хранят информацию о классе эффекта, уровне, длительности, периоде, текущее количество стеков эффекта, захваченные атрибуты, теги для выдачи, asset теги, handler контекста эффекта, в котором указано кто создатель этого GameplayEffectSpec.

Эффекты могут собираться в стек. Тогда вместо простого создания нового экземпляра GameplayEffectSpec, просто увеличивается их количество. Это работает только для длительных и бесконечных эффектов. Есть два типа объединения в стек: по источнику (собираются GameplayEffectSpec от одного источника) и по цели (собираются GameplayEffectSpec одной цели).

Эффекты обычно применяют/удаляют через вызов функции ApplyGameplayEffectSpecToSelf() / RemoveActiveEffects(). Есть три политики удаления эффектов: прервать способность сразу (способность удаляется вместе с удалением соответствующего эффекта), удалить способность после ее окончания (способность завершается и затем удаляется) и ничего не делать (способность никак не реагирует на удаление эффекта).

Модификаторы изменяют атрибуты и являются единственным способом точно сделать это. У эффекта может быть 0 или более модификаторов, каждый изменяет только 1 атрибут через специальную операцию: суммирование, умножение, деление (делит результат модификатора на значение атрибута) или переопределение (перезапись результата в атрибут). Они бывают разных типов: FScalableFloat позволяет задать Data Table или единственное число; модификаторыr, основанные на атрибуте могут использовать текущее или базовое значение атрибута, а также осуществить его захват; собственный класс вычислений, являющийся наиболее гибким вариантом, так как разработан для своих нужд; SetByCaller модификатор позволяют установить значение в runtime и эти значения записываются в GameplayEffectSpec. Теги источника и теги цели можно установить на каждом модификаторе.

Классы собственных требований для применения эффекта (CAR) дают больший контроль над применением эффекта, чем проверка тегов. Они могут быть реализованы в блупринтах и C++, переопределив CanApplyGameplayEffect().

1.5. Компоненты GameplayAbility и Ability Task

Способности (GameplayAbility) – любое действие или навык, которые Actor может выполнять в игре (бег, прыжок, сотворение заклинания, пассивная способность, действующая с определенным периодом, подбор предметов). Активными могут быть несколько способностей одновременно. Способности можно разрабатывать и в блупринтах, и в C++. Сложные способности могут быть реализованы, используя объединение нескольких способностей.

Активация способности происходит следующим образом. Сначала клиент вызывает функцию TryActivateAbility(), происходит вызов InternalTryActivateAbility(). После вызывается CanActivateAbility(), где происходит проверка соответствия тегов, перезарядки, доступность цены и нет ли других активных экземпляров. Если выполнены все условия для активации, то вызывается CallServerTryActivateAbility() и передается сгенерированный ключ-прогноз. Затем - CallActivateAbility(), PreActivate() для инициализации и ActivateAbility(), активирующая способность.

ASC позволяет связывать действия ввода и выданные способности. Действия автоматически активируют эти способности, когда нажимаются кнопки и только если все требования выполнены. Булевое значение bActivateOnInput определяет, будет ли способность активирована сразу, также стоит переопределить функцию AbilityLocalInputPressed() из UAbilitySystemComponent. Дополнительно ASC принимает обычный ввод Confirm и Cancel. Они используются, например, для подтверждения TargetActor или их отмены.

Выдача способности работает также, как и применение эффекта. Создается GameplayAbilitySpec способности и добавляется в ActivatableAbilities. GameplayAbilitySpec содержит информацию о классе, уровне, кнопках ввода, объекте-источнике, выдавшем эту способность; он создается сервером и не реплицируется на другие клиенты.

Существует 3 политики при создании экземпляров способностей. Один экземпляр на Actor – каждый ASC имеет только 1 экземпляр способности, который используется повторно при активации; такой подход используется чаще всего. Новый экземпляр на каждую активацию имеет хуже производительность, но можно заново установить переменные при каждой новой активации. И способности без экземпляров – функционируют через объект класса способности, имеют лучшую производительность, но не могут хранить переменных или привязываться к делегатам и обычно используются для часто вызываемых способностей.

Активировать способности можно с помощью тегов, класс способности, GameplayAbilitySpecHandler или через событие.

У способностей есть несколько нереплицируемых контейнеров тегов с встроенной логикой:

1. Теги способности – теги для описания данной способности.
2. Отмена способности с тегами – способности, имеющие указанные в этом контейнере теги, будут отменены при активации данной способности.
3. Блокировать способности с тегами – не позволяет активироваться способностям, имеющим эти теги.
4. Теги, выдающиеся во время активности способности.
5. Теги, необходимые для активации способности. Нужно иметь все для активации.
6. Теги, блокирующие активацию. Если есть хоть один, то способность не будет активирована.
7. Теги источника, необходимые для активации. Могут быть заданы только если способность вызвана через событие.
8. Блокирующие активацию теги источника. Задаются при вызове через событие.
9. Теги цели, необходимые для активации. Задаются при вызове через событие.
10. Блокирующие активацию теги цели. Задаются при вызове через событие.

Обычно жизненный цикл способности выглядит так: активация, генерация данных, применение и завершение. Но иногда нужно не генерировать данные, а работать с уже существующими. Для этого в GAS есть несколько возможностей. Например, активация способности через событие с передачей данных. Или после активации локальной или серверной способности можно использовать WaitGameplayEvent для ожидания события с данными. Можно использовать структуру TargetData для передачи данных между клиентом и сервером. Или хранить данные на OwnerActor или AvatarActor. Последний способ самый гибкий и может работать даже с способностями, привязанными к вводу пользователя, но нет гарантии, что данные будут синхронизированы при активации способности.

Способности могут иметь стоимость и перезарядку. Цена – количество атрибута, которое нужно иметь, чтобы произошла активация способности. Реализована через мгновенный эффект с одним или несколькими модификаторами, вычитающими цену из атрибутов. Перезарядка – это таймер, который не позволяет реактивировать способность, пока

он не истечет. Обычно он реализован через длительный эффект. После активации способность может в любой момент зафиксировать цену и перезарядку.

Для повышения уровня способности есть два способа. Первый – отменить способность и выдать ее с новым уровнем. Второй – на сервере повысить уровень в GameplayAbilitySpec и пометить его для репликации на клиенте. Первый метод деактивирует способность при ее отмене, второй – нет.

GameplayAbilitySet – UDataAsset классы, хранящие связки с вводом способностей и список начальных способностей для персонажей (Character) с логикой для их выдачи. Подклассы GameplayAbilitySet могут включать дополнительную логику и переменные.

Политика сетевой безопасности способностей определяет, где в сети должна выполняться способность. Она обеспечивает защиту от запуска клиентом недопустимых способностей.

Есть 4 вида: клиент или сервер (никаких ограничений), выполнение только на сервере (только сервер может запустить способность, но клиент может запросить ее отключение), отключение только на сервере (клиенты могут просить запуск способности, но не ее отключение), только на сервере (сервер включает и отключает способности, все запросы клиента игнорируются).

Для внешнего завершения способности у ASC есть несколько функций. Для отключения внутри самой способности можно вызвать CancelAbility(), которое затем вызовет EndAbility() и установит параметр WasCancelled = true.

Способности выполняются за один фрейм. Для совершения действий, которые происходят дольше или требуют ответа на вызов делегата, используются AbilityTask. В GAS есть несколько AbilityTasks изначально: передвижение персонажей, сложные прыжки, бег, используя источники Root Motion, связанные с компонентом движения персонажа; задачи для проигрывания анимационных монтажей, ответ на изменение атрибутов, ответ на изменение эффектов или ответ на ввод пользователя и многое другое. AbilityTasks могут иметь функцию Tick(). Одновременно может быть запущено максимально 1000 параллельных AbilityTask.

Часто создаются собственные AbilityTask в C++. Они состоят из: статичной функции, создающей новые экземпляры AbilityTask; переменных и внутренних вспомогательных функций; функция Activate(), начинающая работу и устанавливающая базовые настройки; делегатов, которые вызываются в конце выполнения; функция OnDestroy() для очистки после завершения AbilityTask.

1.6. Компонент Gameplay Cue

GameplayCue(GC) – используются для воспроизведения звуковых и particle эффектов, тряски камеры, иногда для запуска анимаций. GC вызываются при помощи отправки соответствующего тега вида GameplayCue.ТегА.ТегБ… а также события в GameplayCueManager через ASC. GameplayCueNotify объекты и другие Actor, реализующие IGameplayCueInterface, могут подписываться на эти события. GC обычно реплицируются и прогнозируются.

Есть 3 типа событий: исполнить (Execute), добавить (Add), удалить (Remove). Также существуют 2 класса GameplayCueNotify: статичный (GameplayCueNotify_Static) и GameplayCueNotify_Actor. Статичный класс отвечает на событие исполнить и может быть вызван эффектом мгновенным или периодичным; идеально подходит для разовых эффектов. GameplayCueNotify_Actor отвечает на события добавить и удалить, вызываются длительными

и бесконечными эффектами; подходит для повторяющихся звуков или частиц, можно добавить несколько GC за раз.

Локальные GC – GC, которые исполняются, добавляются и удаляются только на индивидуальном клиенте. Обычно используются при столкновениях снарядов или в ближнем бою или GC, которые отправлены из анимационных монтажей.

Выполнение всех GC на ASC компоненте может быть выключено, установив bSuppressGameplayCues = true. Также можно отправлять события GC на компонент ASC актора цели или актора источника вручную в GameplayEffectExecutionCalculations, вызвав OutExecutionOutput.MarkGameplayCuesHandledManually().

1.7. GameplayPrediction или прогноз событий

GAS поддерживает прогноз событий на клиенте. Прогнозировать можно только то, что можно откатить, то есть активацию способностей, запуск событий, применение эффектов, изменения атрибутов через модификаторы, изменение тегов, вызов событий GameplayCue, монтажи и передвижение. Снятие эффектов и периодичные эффекты не прогнозируются. Прогноз на клиенте в GAS означает, что клиенту не нужно ждать одобрения сервера, чтобы совершить действие. Предполагается, что сервер даст согласие, а также прогнозируются цели для этих действий. Сервер позже проверяет предположение клиента, и в случае ошибки откатывает состояние к тому, которое соответствует состоянию на сервере.

1.8. Компоненты Target Data и Target Actor

FGameplayAbilityTargetData – общая структура для данных о целях, предназначенная для отправления по сети между сервером и клиентом. В TargetData (TD) обычно содержатся ссылки на Actor или объекты, FHitResult, а также данные о локации, направлении, источнике информации. Можно и даже рекомендуется создать свой производный класс и хранить в нем любую нужную информацию.

TD обычно создается Target Actor или вручную и используется в задачах способностей или эффектах через контекст эффектов. Обычно напрямую FGameplayAbilityTargetData не передается, вместо этого используется FGameplayAbilityTargetDataHandle, содержащий массив с указателями на FGameplayAbilityTargetData. Такой подход обеспечивает поддержку полиморфизма TargetData.

Target Actor базируются на обычных Actor, то есть они могут иметь какой-нибудь визуальный компонент для показа куда и на что они целятся, например, статичные меши (показать расположение при создании объекта в мире) или decals (показать область воздействия эффекта).

Target Actor могут дополнительно использовать GameplayAbilityWorldReticles для показа текущих целей. Они являются Actor, поэтому могут иметь визуальные компоненты.

Заключение

В статье рассмотрены основные компоненты плагина Gameplay Ability System для Unreal Engine и их особенности, и показано, где и как они используются.

Литература

1. Официальная документация по GameplayAbilitySystem от Unreal Engine. – Режим доступа:https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/GameplayAbilitySystem/. – (Дата обращения: 20.03.2023).

2. Неофициальная документация по GameplayAbilitySystem – Режим доступа: https://github.com/tranek/GASDocumentation – (Дата обращения: 10.04.2023).

3. Gameplay Ability System: как мы используем плагин для разработки Survival RPG. – Режим доступа: https://habr.com/ru/company/socialquantum/blog/505886/ – (Дата обращения: 01.04.2023).

УДК: 519.61

РАЗЛОЖЕНИЕ ФУНКЦИИ ГРИНА ЗАДАЧИ ОБ ОГРАНИЧЕННЫХ РЕШЕНИЯХ В РЯД ПО ФУНКЦИЯМ ЛАГЕРРА

Е. Д. Хороших

Воронежский государственный университет

Для дифференциального уравнения $x'(t) = Ax(t) + f(t), t \in \mathrm{P}$ с матричным коэффициентом A рассматривается задача о нахождении ограниченного решения x при условии, что свободный член f ограничен. В предположении, что матрица A не имеет собственных значений на мнимой оси, предложен алгоритм численного нахождения функции Грина, основанный на ее приближении функциями Лагерра. При этом на левой и правой полуосях берутся функции Лагерра с разными экспоненциальными множителями $e^{-t/2}$ и $e^{t/2}$.

Введение

Пусть A — квадратная матрица. Рассмотрим дифференциальное уравнение

$$x'(t) = Ax(t) + f(t), \qquad t \in \mathrm{P}. \tag{1}$$

Задачей об ограниченных решениях для этого уравнения называют задачу о нахождении ограниченного на P решения x в предположении, что f непрерывна и ограничена. Задача об ограниченных решениях для разных типов уравнений изучалась в [4-6, 8, 9, 12, 14, 18, 22] и многих других работах. Хорошо известно, что эта задача однозначно разрешима тогда и только тогда, когда спектр матрицы A не пересекает мнимую ось. В этом случае решение может быть выражено через функцию Грина Γ:

$$x(t) = \int_{-\infty}^{\infty} \Gamma(s) f(t-s) ds.$$

Настоящая работа посвящена численному нахождению функции Грина Γ. Ранее задача численного нахождения функции Грина обсуждалась в [4, 19]. Поскольку матричные элементы функции Грина представляют собой линейные комбинации экспоненциальных функций, умноженных на многочлены, мы предлагаем вычислять функцию Грина путем ее разложения по ортонормированному базису в $L_2(\mathrm{P})$, состоящему из функций Лагерра, которые имеют аналогичный вид. Как известно, функции Лагерра образуют ортонормированный базис на полуоси. Поэтому фактически мы используем два семейства функций Лагерра — для левой и правой полуосей.

На практике посчитать полный ряд Фурье функции Грина не удается — с разумной точностью, как правило, можно лишь посчитать около 30 слагаемых на каждой из полуосей. Чтобы повысить точность вычислений, в настоящей работе предлагается применять не классические функции Лагерра, т.е. использующие не экспоненциальный множитель $e^{\pm t/2}$, а использующие «растянутые» экспоненциальные множители $e^{-t/2}$ и $e^{t/2}$. Численные эксперименты показывают, что это существенно увеличивает точность. При этом, чем дальше

действительная часть собственного значения λ от соответствующего ⌉, тем медленнее сходится разложение функции $e^{\lambda t}$ в ряд Фурье по функциям Лагерра с соответствующим ⌉. Чтобы добиться одинакового качества сходимости ко всем функциям $e^{\lambda t}$, входящим в функцию Грина, предлагается следующий алгоритм выбора ⌉$^{\pm}$ (и это является основной идеей настоящей работы). Собственные значения матрицы A делятся на две части — на лежащие в левой полуплоскости (они обозначаются $\sigma^{+}(A)$, поскольку с ними связано поведение функции Грина на правой полуоси) и на лежащие в правой полуплоскости (они обозначаются $\sigma^{-}(A)$, поскольку с ними связано поведение функции Грина на левой полуоси). В качестве ⌉$^{-=}$ берется среднее значение действительных частей чисел $\lambda \in \sigma^{-}(A)$, а точнее, под средним значением понимается удвоенное среднее геометрическое $2 \cdot \sqrt{Re\lambda_{\min} \cdot Re\lambda_{\max}}$ наименьшей и наибольшей из действительных частей чисел $\lambda \in \sigma^{-}(A)$. Такой выбор ⌉$^{-=}$ обеспечивает минимальное возможное отношение $Re\lambda/$⌉$^{-}$ (или ⌉$^{-}/Re\lambda$) по всем $\lambda \in \sigma^{-}(A)$. Численные эксперименты показывают, что такой выбор ⌉$^{-=}$ является близким к оптимальному. Аналогично выбирается ⌉$^{+=}$.

Преобразованием Фурье $\hat{\Gamma}$ функции Грина является сужение $\omega \mapsto (i\omega I - A)^{-1}/\sqrt{2\pi}$ резольвенты на мнимую ось. Значение этой функции в любой точке ω эффективно вычисляется. Раскладывая $\hat{\Gamma}$ по базису из преобразований Фурье функций Лагерра, мы находим коэффициенты c_k разложения функции Грина Γ по исходному базису. Таким образом, задача вычисления c_k сводится к вычислению некоторых интегралов. Одним из преимуществ используемого метода является то, что он позволяет оценить точность приближения.

Ниже описываются минимальные теоретические сведения, предлагаемый алгоритм и результаты численного эксперимента.

1. Задача об ограниченных решениях

Пусть $b \in \mathrm{N}$. Введем в пространстве X^b какую-нибудь норму; напомним [17, с.~162], что любые две нормы на конечномерном пространстве эквивалентны. Обозначим через $C = C(\mathrm{P}, \mathrm{X}^b)$ линейное пространство всех непрерывных ограниченных функций $f : \mathrm{P} \to \mathrm{X}^b$ с нормой

$$\|f\| = \|f\|_C = \sup_{t \in \mathrm{P}} \|f(t)\|.$$

Обозначим через $C^1 = C^1(\mathrm{P}, \mathrm{X}^b)$ линейное пространство всех непрерывно дифференцируемых функций $f : \mathrm{P} \to \mathrm{X}^b$, ограниченных вместе с производной, с нормой

$$\|f\| = \|f\|_{C^1} = \max\{\|f\|_C, \|f'\|_C\}.$$

Нетрудно показать, что пространства C и C^1 банаховы.

Обозначим через $\mathrm{X}^{b \times b}$ множество всех комплексных матриц размера $b \times b$. Пусть

$A \in \mathrm{X}^{b\times b}$. Рассмотрим дифференциальное уравнение

$$x'(t) = Ax(t) + f(t), \qquad t \in \mathrm{P}. \tag{2}$$

Задачей об ограниченных решениях для уравнения (2) называют задачу о нахождении решения $x \in C^1$ в предположении, что $f \in C$.

Теорема 1 [6, с. 119, теорема 4.1]. *Пусть A — квадратная комплексная матрица. Уравнение (2) имеет единственное решение $x \in C^1$ для любой функции $f \in C$ тогда и только тогда, когда спектр $\sigma(A)$ матрицы A не пересекает мнимую ось. В этом случае существует такая функция $\Gamma : \mathrm{P} \to \mathrm{X}^{b\times b}$, что решение уравнения (2) допускает представление*

$$x(t) = \int_{-\infty}^{\infty} \Gamma(s) f(t-s) ds.$$

Функцию Γ называют [6] *функцией Грина* задачи об ограниченных решениях для уравнения (2). Функция Грина является непрерывной (и даже бесконечно дифференцируемой) всюду, кроме точки $t = 0$. Известно [6, с. 49, 119], что функция Γ экспоненциально убывает на бесконечности: существуют такие $N, \nu > 0$, что

$$\|\Gamma(t)\| \le Ne^{-\nu t}. \tag{3}$$

Здесь $\|\cdot\|$ — произвольная норма в пространстве $\mathrm{X}^{b\times b}$ (в силу конечномерности пространства $\mathrm{X}^{b\times b}$ все нормы эквивалентны). Более точные оценки функции Грина можно найти в [2, 3, 5, 20, 21].

Будем использовать преобразование Фурье с нормировкой

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t) e^{-i\omega t} dt, \qquad \omega \in \mathrm{P}.$$

При такой нормировке преобразование Фурье является изометрическим изоморфизмом пространства $L_2 = L_2(\mathrm{P}, \mathrm{X})$ на себя.

Предложение 2. *Функция Грина Γ является обратным преобразованием Фурье функции*

$$\omega \mapsto \frac{1}{\sqrt{2\pi}} (i\omega I - A)^{-1}. \tag{4}$$

Доказательство. Перейдем к преобразованию Фурье в формуле (2):

$$i\omega \hat{x}(\omega) = A\hat{x}(\omega) + \hat{f}(\omega).$$

Отсюда

$$\hat{x}(\omega) = (i\omega I - A)^{-1} \hat{f}(\omega).$$

Переходя к обратному преобразованию Фурье (вспоминая, что мы пользуемся нормированным преобразованием Фурье) и используя формулу преобразования Фурье свертки, получаем

$$x(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \mathrm{P}(s) f(t-s) ds,$$

где P — обратное преобразование Фурье функции $\omega \mapsto (i\omega I - A)^{-1}$. Отсюда видно, что можно положить $\Gamma(s) = \frac{1}{\sqrt{2\pi}} \mathrm{P}(s)$.

2. Алгоритм вычислений

Стандартизированным многочленом Лагерра называют [15, с. 219], [16, с. 71] многочлен вида

$$L_n(x) = \frac{1}{n!} e^x [x^n e^{-x}]^{(n)}, \qquad n = 0,1,\ldots, \ x \geq 0.$$

Отождествим пространство $L_2(0,+\infty)$ с подпространством пространства $L_2(\mathrm{P})$, состоящим из функций, равных нулю на $(-\infty,0)$. Аналогичным образом, отождествим пространство $L_2(-\infty,0)$ с подпространством пространства $L_2(\mathrm{P})$, состоящим из функций, равных нулю на $(0,+\infty)$. Очевидно, пространство $L_2(\mathrm{P})$ представимо в виде прямой суммы

$$L_2(\mathrm{P}) = L_2(-\infty,0) \oplus L_2(0,+\infty).$$

Функциями Лагерра называют функции

$$l_n^+(x) = \begin{cases} e^{-x/2} L_n(x) & \text{при } x > 0, \\ 0 & \text{при } x < 0, \end{cases} \qquad n = 0,1,\ldots,$$

$$l_n^-(x) = \begin{cases} 0 & \text{при } x > 0, \\ e^{x/2} L_n(-x) & \text{при } x < 0, \end{cases} \qquad n = 0,1,\ldots.$$

Известно [7, с. 404], [13, с. 59], [1, с. 67], что функции l_n^+ образуют ортонормированный базис в пространстве $L_2(0,+\infty)$, а функции l_n^- образуют ортонормированный базис в пространстве $L_2(-\infty,0)$. Поэтому совокупность функций l_n^+ и l_n^- образует ортонормированный базис в $L_2(\mathrm{P})$.

Возьмем произвольные числа $\gamma^+ > 0$ и $\gamma^- > 0$. *Растянутыми функциями Лагерра* назовем функции

$$l_n^+(x,\gamma^+) = \sqrt{\gamma^+}\, l_n^+(\gamma^+ x), \qquad n = 0,1,\ldots,$$

$$l_n^-(x,\gamma^-) = \sqrt{\gamma^-}\, l_n^-(\gamma^- x), \qquad n = 0,1,\ldots.$$

Очевидно, набор этих функций также образует ортонормированный базис в пространстве $L_2 = L_2(\mathrm{P},\mathrm{X})$.

Функции $l_n^\pm$ похожи на типичный график функции Грина. Поэтому, чтобы приближенно вычислить функцию Грина, будем искать коэффициенты $c_n^\pm \in \mathrm{X}^{b\times b}$ разложения Γ по этому базису (в силу оценки (3) функция Грина, очевидно, принадлежит $L_2(\mathrm{P})$):

$$\Gamma = \sum_{n=0}^{\infty} c_n^+ l_n^+ + \sum_{n=0}^{\infty} c_n^- l_n^-. \qquad (5)$$

Поскольку функцию Грина Γ мы не знаем, перейдем к преобразованию Фурье. В силу предложения 2 имеем $\hat{\Gamma}(\omega) = (i\omega I - A)^{-1}/\sqrt{2\pi}$, что легко численно находится в любой точке ω. Поэтому вместо использования разложения (5) будем искать коэффициенты $c_n^\pm$, исходя из разложения

$$\hat{\Gamma}(\omega) = \sum_{n=0}^{\infty} c_n^+ \hat{l}_n^+(\omega,\gamma^+) + \sum_{n=0}^{\infty} c_n^- \hat{l}_n^-(\omega,\gamma^-).$$

Вычисления показывают, что преобразования Фурье функций $l_n^{\pm}$ имеют вид

$$\hat{l}_n^+(\omega,\rceil^+) = -i^{n+1}\sqrt{\frac{2\rceil^+}{\pi}}(\rceil^+ - 2i\omega)^n(2\omega - i\rceil^+)^{-n-1},$$

$$\hat{l}_n^-(\omega,\rceil^-) = -(-i)^{n+1}\sqrt{\frac{2\rceil^-}{\pi}}(\rceil^- + 2i\omega)^n(2\omega + i\rceil^-)^{-n-1}.$$

Очевидно, эти две функции являются комплексно сопряженными. Поэтому нахождение $c_n^{\pm}$ сводится к вычислению интегралов

$$\begin{aligned} c_n^+ &= \int_{-\infty}^{+\infty}(i\omega I - A)^{-1}\overline{\hat{l}_n^+(\omega,\rceil^+)}d\omega = \int_{-\infty}^{+\infty}(i\omega I - A)^{-1}\hat{l}_n^-(\omega,\rceil^+)d\omega, \\ c_n^- &= \int_{-\infty}^{+\infty}(i\omega I - A)^{-1}\overline{\hat{l}_n^-(\omega,\rceil^-)}d\omega = \int_{-\infty}^{+\infty}(i\omega I - A)^{-1}\hat{l}_n^+(\omega,\rceil^-)d\omega. \end{aligned} \tag{6}$$

Конечно, реальные вычисления состоят в замене рядов частичными суммами:

$$\Gamma \approx \sum_{n=0}^{m_1} c_n^+ l_n^+ + \sum_{n=0}^{m_2} c_n^- l_n^-. \tag{7}$$

Для оценки точности вычислений использовалось тождество наименьших квадратов [10, следствие 9.8]

$$\left\| \Gamma - \sum_{n=0}^{m_1} c_n^+ l_n^+ - \sum_{n=0}^{m_2} c_n^- l_n^- \right\|^2 = \|\Gamma\|_{L_2}^2 - \sum_{n=0}^{m_1}\left\|c_n^+\right\|^2 - \sum_{n=0}^{m_2}\left\|c_n^-\right\|^2 = \left\|\hat{\Gamma}\right\|_{L_2}^2 - \sum_{n=0}^{m_1}\left\|c_n^+\right\|^2 - \sum_{n=0}^{m_2}\left\|c_n^-\right\|^2.$$

Чтобы вычисления по этой формуле проходили быстрее, для матриц использовалась норма Фробениуса

$$\|A\|_F = \sqrt{\sum_{i=1}^{b}\sum_{j=1}^{b}|a_{ij}|^2}.$$

3. Выбор | и численный эксперимент

На каждой из полуосей $\rceil^{\pm}$ выбиралось независимо, но по единому правилу.

Вначале выполнялся эксперимент по «совместимости» собственных значений с коэффициентом растяжения | . Для этого тестовая функция (соответствующая собственному значению $\lambda = 1/2$)

$$u(x) = \begin{cases} 0 & \text{при } x > 0, \\ e^{x/2} & \text{при } x < 0, \end{cases}$$

приближалась 30-й частичной суммой ряда Фурье по функциям Лагерра $l_n^-(\cdot,\rceil^-)$ с различными $\rceil^-$. Для $\rceil^- \in [1/10, 10]$ частичная сумма и тестовая функция оказались визуально неразличимы. Для $\rceil^- \in [1/30, 30]$ погрешность доходит примерно до 5%. Из этого мы сделали вывод, что отклонение $\rceil^-$ от собственного значения в 10 раз (в ту или иную сторону) допустимо. Более того, отклонение в 30 раз также в исключительных случаях приемлемо. Чтобы сделать подобные отклонения для всех собственных значений одинаковыми, был принят следующий алгоритм выбора $\rceil^-$, что является основной идеей настоящей работы. Обозначим через $\sigma^-(A)$ собственные значения матрицы A, лежащие в правой полуплоскости. В качестве $\rceil^-$ бралось удвоенное среднее геометрическое $2\cdot\sqrt{Re\lambda_{\min}\cdot Re\lambda_{\max}}$ наименьшей и

наибольшей из действительных частей чисел $\lambda \in \sigma^{-}(A)$. Аналогично выбиралось $\rceil^{+=}$.

Проверка эффективности этого способа выбора $\rceil^{\pm}$ проводилась на случайных матрицах A небольшого размера. Вычислялась точность приближения функции Грина с использованием описанного способа выбора $\rceil^{\pm}$ и с использованием оптимального $\rceil^{\pm}$, найденного методом золотого сечения. В худших случаях результаты расходились примерно в полтора раза.

В заключение была написана компьютерная программа на языке пакета «Математика» [23, 11], реализующая описанный алгоритм. Численный эксперимент показал следующее. В качестве матрицы A была взята матрица размера 10×10, элементами которой являлись случайные числа, равномерно распределенные в промежутке $(-1,1)$. Вероятность того, что спектр такой матрицы пересекается с мнимой осью, равна нулю, поэтому по теореме 1 функция Грина существует. В качестве значений m_1 и m_2 бралось число 30. Затем вычислялись числа $c_n^{\pm}$ по формуле (6). Вычислялись значения $\left\|c_n^{\pm}\right\|_F$. Если последнее из этих чисел оказывалось больше 0.01, соответствующее число m_1 или m_2 увеличивалось. Находилась точность по формуле

$$\sqrt{\left\|\hat{\Gamma}\right\|_{L_2}^2 - \sum_{n=0}^{m_1}\left\|c_n^{+}\right\|^2 - \sum_{n=0}^{m_2}\left\|c_n^{-}\right\|^2}.$$

Среднее значение точности получалось чуть больше 10^{-3}.

Литература

1. Ахиезер, Н. Лекции по интегральных преобразованиях / Н.И. Ахиезер. Харьков : Вища школа. Изд-во при Харьк. ун-те, 1984. – 120 с.

2. Баскаков, А. Г. Оценки функции Грина и параметров экспоненциальной дихотомии гиперболической полугруппы операторов и линейных отношений / А. Г. Баскаков // Математ. сб. – 2015. – Т. 206, № 8. – С. 23–62.

3. Булгаков, А. Я. Оценка матрицы Грина, непрерывность параметра дихотомии. / А.Я. Булгаков // Сиб. Матем. Журн. – 1989. – Т. 30, № 1. – С. 178–182.

4. Годунов, С. К. Обыкновенные дифференциальные уравнения с постоянными коэффиц иентами : учебное пособие / С. К. Годунов. – Новосибирск : Издательство Новосибирского университета, 1994. – Т. 1: Краевые задачи. – 264 с.

5. Годунов, С. К. Матрица Грина краевой задачи для обыкновенных дифференциальных уравнений / С. К. Годунов, В. М. Гордиенко // Успехи математических наук. – 1984. – Т. 39, № 1(235). – С. 39–76.

6. Далецкий, Ю. Л. Устойчивость решений дифференциальных уравнений в банаховом пространстве / Ю. Л. Далецкий, М. Г. Крейн. Нелинейный анализ и его приложения. – М. : Наука, 1970. – 536 с.

7. Колмогоров, А. Н. Элементы теории функций и функционального анализа / А. Н. Колмогоров, С. В. Фомин. – М. : Наука, 1972. – 496 с.

8. Курбатов, В. Г. Об ограниченных решениях дифференциально-разностных уравнений / В. Г. Курбатов // Сиб. матем. журн. – 1986. – Т. 27, № 1. – С. 68–79.

9. Курбатов, В. Г. О функции Грина дифференциального-разностного уравнения / В.Г. Курбатов // Дифференц. уравнения. – 1987. – Т. 24, № 3. – С. 525–527.

10. Курбатов, В. Г. Алгебра : учебное пособие / В. Г. Курбатов. – Воронеж :

Издательский дом ВГУ, 2022. – 604 с.

11. Курбатов, В. Г. Пакет “Математика” в прикладных научных исследованиях : учебное пособие / В. Г. Курбатов, В. Е. Чернов. – Воронеж : Издательский дом ВГУ, 2016. – 240 с.

12. Курбатова, И. В. Представление возмущенной функции Грина задачи об ограниченных решениях / И. В. Курбатова // Вестник Воронеж. гос. ун-та. Серия: Физ.-мат. науки. – 2020. – № 1. – С. 67–74–152.

13. Никифоров, А. Ф. Специальные функции математической физики / А. Ф. Никифоров, В. Б. Уваров. – Второе . – М. : Наука, 1985. – 344 с.

14. Печкуров, А. В. Бисекториальные операторные пучки и задача об ограниченных решениях / А. В. Печкуров // Известия вузов. Математика. – 2012. – № 3. – С. 31–41.

15. Суетин, П. К. Классические ортогональные многочлены / П. К. Суетин. – Второе . М. : Наука, 1979. – 416 с.

16. Функции математической физики / Ж. Кампе де Ферье, Р. Кемпбелл, Г. Петьо, Т. Фогель. – М. : ГИМФЛ, 1974. – 104 с.

17. Хелемский, А. Я. Лекции по функциональному анализу / А. Я. Хелемский. – М. : МЦНМО, 2004. – 552 с.

18. Chicone, Evolution semigroups in dynamical systems and differential equations / C. Chicone, Y. Latushkin. – Providence, RI : American Mathematical Society, 1999. – Vol. 70 of Mathematical Surveys and Monographs. – x+361 p.

19. Kurbatov, V. G. Computation of Green’s function of the bounded solutions problem / V.G. Kurbatov, I. V. Kurbatova // Comput. Methods Appl. Math. – 2018. – Vol. 18, no. 4. – P. 673–685.

20. Kurbatov, V. G. The Gelfand–Shilov type estimate for Green’s function of the bounded solutions problem / V. G. Kurbatov, I. V. Kurbatova // Qual. Theory Dyn. Syst. – 2018. – Oct. – Vol. 17, no. 3. – P. 619–629.

21. Kurbatov, V. G. Green’s function of the problem of bounded solutions in the case of a block triangular coefficient / V. G. Kurbatov, I. V. Kurbatova // Operators and Matrices. – 2019. Vol. 13, no. 4. – P. 981–1001.

22. Kurbatova, I. V. Representations of Green’s function of the bounded solutions problem for a differential-algebraic equation / I. V. Kurbatova, A. V. Pechkurov // Banach J. Math. Anal. – 2020. – Vol. 14, no. 3. – P. 707–736.

23. Wolfram, S. The Mathematica book / S. Wolfram. – Fifth edition. – New York : Wolfram Media, 2003. – 1488 p.

УДК 004

РАЗРАБОТКА МЕССЕНДЖЕРА И ПОИСК СЛЕДОВ

А. Д. Худобин

Воронежский государственный университет

Введение

Сервисы для обмена мгновенными сообщениями являются частью современной повседневной жизни, всё больше людей в мире пользуется ими. Жизнь человека становится все более зависимой от мессенджеров. В них люди переписываются, производят видео- и аудиозвонки, отправляют друг другу файлы, обмениваются новостями, проводят обучение. Для использования мессенджеров необходим доступ к сети Интернет.

С развитием популярности мессенджеров повысился риск утечки персональных данных пользователей, получения доступа к перепискам другими людьми. Утечки могут происходить по нескольким причинам: невнимательность разработчиков сервиса обмена мгновенными сообщениями, намеренная продажа данных пользователей разработчиками, невнимательность пользователя (например, передача своих данных третьим лицам).

Следы — это какие-либо данные, которые остаются после использования программ для обмена мгновенными сообщениями. Это сведения о регистрации, входе, выходе из приложения, написании сообщения, списке контактов, фотографии. Мессенджеры оставляют разное количество следов. Пользователям важно выбирать программы для обмена мгновенными сообщениями, которые оставляют как можно меньше следов, чтобы защитить себя.

Актуальность работы обусловлена тем, что сервисы мгновенных сообщений становятся всё популярнее, а персональные данные пользователей и их переписки находятся под угрозой. Поэтому необходимо разработать мессенджер, который обеспечивает защиту данных пользователей. В рамках работы возникают следующие задачи:

1. Выявить и проанализировать следы, которые остаются после использования мессенджеров на персональных компьютерах и в веб-версиях на примере популярных мессенджеров Telegram, Viber и WhatsApp.
2. Произвести получение удаленных сообщений из мобильного приложения Telegram с Android телефона в простых и секретных чатах.
3. Выбрать средства реализации приложения.
4. Реализовать приложение.
5. Провести тестирование приложения.

Требования к приложению:

1. Приложение не должно требовать от пользователей обязательной регистрации.
2. Информация, хранящаяся в базе данных, должна удаляться.

1. Поиск следов на персональном компьютере

Рассмотрим поиск истории сообщений пользователя, хранящейся в персональном компьютере. Данные о переписках, звонках и контактах хранятся в виде файла базы данных, она сохраняется в резервной копии или внутренней памяти мобильного устройства, с помощью

которого аккаунт пользователя привязывается в приложении для персонального компьютера, а информация, содержащаяся там, зашифрована.

Мессенджер Viber. Эта программа дает просматривать переписки пользователей без ключа дешифровки, то есть, чтобы просмотреть данные, нужно открыть файл viber.db, который находится в:

- для Windows: C:\Users\Имя_пользователя\AppData\Roaming\ViberPC\Номер_телефона\viber.db;
- для Ubuntu: ViberPC\Номер_телефона\viber.db.

Откроем файл viber.db в программе DB Browser for SQLite. Сначала посмотрим таблицу Contact. Можем увидеть, что здесь хранятся не только пользователи Viber, но и все остальные контакты пользователя, которые у него есть в телефонной книге. В столбце MID видим, кто из телефонной книги имеет аккаунт в Viber. Таблицу Contact можно посмотреть на рис. 1.

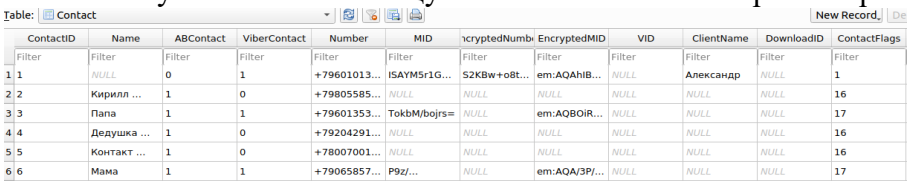


Table: Contact

	ContactID	Name	ABContact	ViberContact	Number	MID	ncryptedNumb	EncryptedMID	VID	ClientName	DownloadID	ContactFlags
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	NULL	0	1	+79601013...	ISAYM5r1G...	S2KBw+o8t...	em:AQAhIB...	NULL	Александр	NULL	1
2	2	Кирилл ...	1	0	+79805585...	NULL	NULL	NULL	NULL	NULL	NULL	16
3	3	Папа	1	1	+79601353...	TokbM/bojrs=	NULL	em:AQBOiR...	NULL	NULL	NULL	17
4	4	Дедушка ...	1	0	+79204291...	NULL	NULL	NULL	NULL	NULL	NULL	16
5	5	Контакт ...	1	0	+78007001...	NULL	NULL	NULL	NULL	NULL	NULL	16
6	6	Мама	1	1	+79065857...	P9z/...	NULL	em:AQA/3P/...	NULL	NULL	NULL	17

Рис. 1. Файл viber.db, таблица Contact

Теперь посмотрим таблицу Viber EventInfo. Здесь хранятся все сообщения, стикеры, информация прочитано сообщение или нет, геолокация, информация о том, кто отправил сообщение, время отправки каждого сообщения. Таблицу EventInfo можно увидеть на рис. 2.

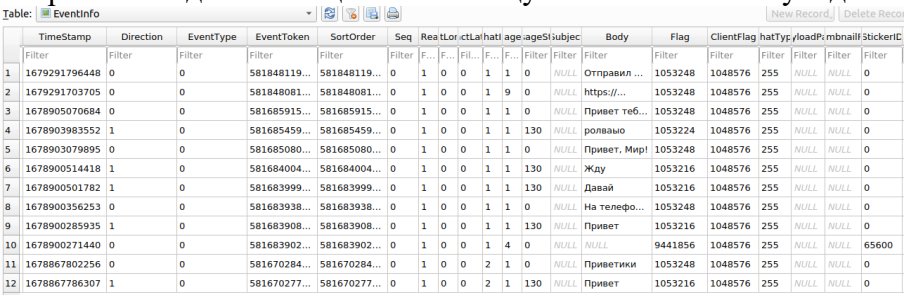


Table: EventInfo

	TimeStamp	Direction	EventType	EventToken	SortOrder	Seq	Rea	tLor	ctLa	hatl	age	ageSt	subjec	Body	Flag	ClientFlag	hatTyp	loadP	mbnailF	StickerID
	Filter	Filter	Filter	Filter	Filter	Filter	F...	F...	Fil...	F...	F...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1679291796448	0	0	581848119...	581848119...	0	1	0	0	1	1	0	NULL	Отправил ...	1053248	1048576	255	NULL	NULL	0
2	1679291703705	0	0	581848081...	581848081...	0	1	0	0	1	9	0	NULL	https://...	1053248	1048576	255	NULL	NULL	0
3	1678905070684	0	0	581685915...	581685915...	0	1	0	0	1	1	0	NULL	Привет теб...	1053248	1048576	255	NULL	NULL	0
4	1678903983552	1	0	581685459...	581685459...	0	1	0	0	1	1	130	NULL	ролваыо	1053224	1048576	255	NULL	NULL	0
5	1678903079895	0	0	581685080...	581685080...	0	1	0	0	1	1	0	NULL	Привет, Мир!	1053248	1048576	255	NULL	NULL	0
6	1678900514418	1	0	581684004...	581684004...	0	1	0	0	1	1	130	NULL	Жду	1053216	1048576	255	NULL	NULL	0
7	1678900501782	1	0	581683999...	581683999...	0	1	0	0	1	1	130	NULL	Давай	1053216	1048576	255	NULL	NULL	0
8	1678900356253	0	0	581683938...	581683938...	0	1	0	0	1	1	0	NULL	На телефо...	1053248	1048576	255	NULL	NULL	0
9	1678900285935	1	0	581683908...	581683908...	0	1	0	0	1	1	130	NULL	Привет	1053216	1048576	255	NULL	NULL	0
10	1678900271440	0	0	581683902...	581683902...	0	1	0	0	1	4	0	NULL	NULL	9441856	1048576	255	NULL	NULL	65600
11	1678867802256	0	0	581670284...	581670284...	0	1	0	0	2	1	0	NULL	Приветики	1053248	1048576	255	NULL	NULL	0
12	1678867786307	1	0	581670277...	581670277...	0	1	0	0	2	1	130	NULL	Привет	1053216	1048576	255	NULL	NULL	0

Рис. 2. Файл viber.db, таблица EventInfo

Время совершения звонка в столбце TimeStamp представлено в формате Unix-time. Этот формат зависит от количества секунд, которые прошли с 1 января 1970 года до настоящего времени. Проведем декодирование времени, для этого нужно убрать последние три цифры из timestamp. Проведем декодирование времени первого сообщения в таблице EventInfo. Результат декодирования времени находится на рис. 3.

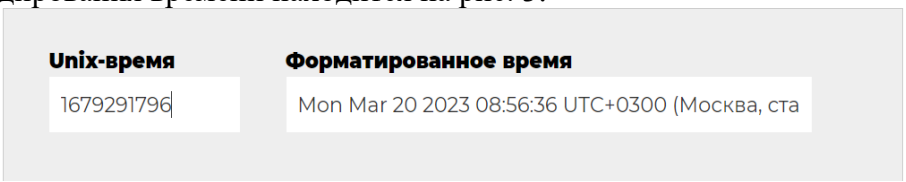


Рис. 3. Декодирование времени, произведено на сайте https://time.is/

Информация о звонках находится в таблице Calls. В ней есть поля Duration — это продолжительность звонка в секундах, Type — вид звонка, который совершает пользователь.

Информация о WhatsApp на персональном компьютере находится в C:\Users\Имя_пользователя\AppData\Roaming\WhatsApp\Cache. Здесь есть файл data2, в котором мы можем получить фотографии профилей всех контактов. Для этого воспользуемся программой ThumbnailExpert, которая используется, чтобы получать кэш программ.

Информацию о номерах пользователей можно получить в папке IndexedDB, в C:\Users\Имя_пользователя\AppData\Roaming\WhatsApp\IndexedDB\file__0.indexeddb.leveldb. Этот файл изображен на рис. 4.

Рис. 4. Номера телефонов в файле 00023

Теперь составим таблицу, в которой укажем, какие следы приложения мессенджеров для персональных компьютеров оставляют на нем. В табл. 1 представлена сравнительная таблица популярных мессенджеров, в которой можно посмотреть, какие виды следов они оставляют.

Таблица 1

Следы приложений мессенджеров для персональных компьютеров

	Viber	WhatsApp	Telegram
Дата и время установки	+	+	+
Дата и время запуска	+	+	+
Загруженные файлы	+	+	+ если загружены
Сообщения	+	-	-
Контакты	+ все контакты телефона	+ номера и аватары	-
Аудио- и видеозвонки	+	-	-

2. Поиск следов на Android

Проведем поиск следов на телефоне с операционной системой Android. Будем концентрироваться на получении удаленных записей в секретных и простых чатах.

Следы на телефоне в Telegram хранятся в файлах cache4.db, каталогах Telegram Images, Telegram и cache [1]. В табл. 2. подробнее опишем, где и что хранится в них.

Таблица 2

Следы Telegram на телефоне с операционной системой Android

Содержимое	Расположение
База данных Telegram	/data/org.telegram.messenger/files/cache.4db
Копии отправленных фотографий	/media/0/Telegram/Telegram Images
Копии исходящих фотографий	/media/0/Pictures/Telegram
Фотографии профилей контактов	/media/0/Android/data/org.telegram.messenger/cache

Для доступа к большинству этих файлов необходимо обладать правами суперпользователя (root) на исследуемом телефоне. Для их получения воспользуемся программой-эмулятором NoxPlayer 7.0.5.5. В настройках выберем пункт «Рут».

Напишем несколько сообщений в простом и секретном чатах, отправим несколько картинок и удалим для обоих пользователей некоторые сообщения.

Откроем базу данных cache4.db в DB Browser for SQLite. Основную информацию в базе содержат таблицы messages_v2, users, enc_chats, chats. Напишем SQL-запрос, в котором объединим таблицы users и messages_v2. В таблице можем заметить, что в секретном чате имя собеседника недоступно, а наше имя зашифровано. SQL-запрос и полученная таблица показаны на рис. 5.

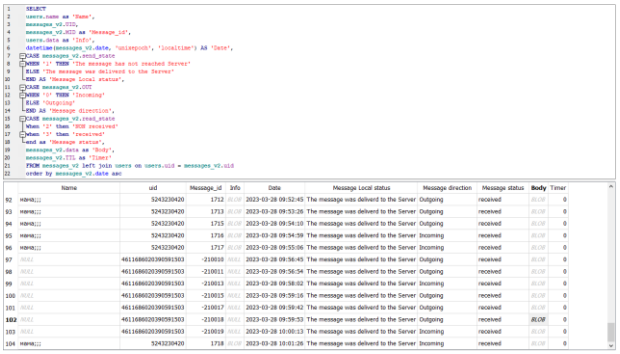


Рис. 5. Таблица с сообщением и отправителем

Столбец Body помечен надписью BLOB, это означает, что данные хранятся в двоичном виде. В DB Browser for SQLite можно их перевести. Ячейку базы данных можно увидеть на рис. 6.

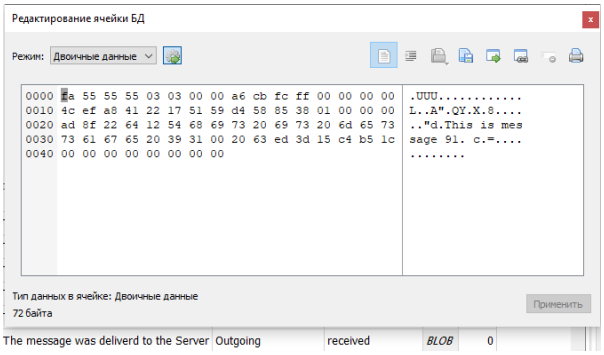


Рис. 6. Двоичные данные

Мы можем увидеть, что в DB Browser for SQLite нет доступа к удаленным сообщениям. Попробуем открыть файл cache4.db-wal в программе FQLite Carving Tool [2]. В таблице messages_v2 можем заметить все удаленные сообщения и из простого чата (рис. 7.), и из секретного (рис. 8.).

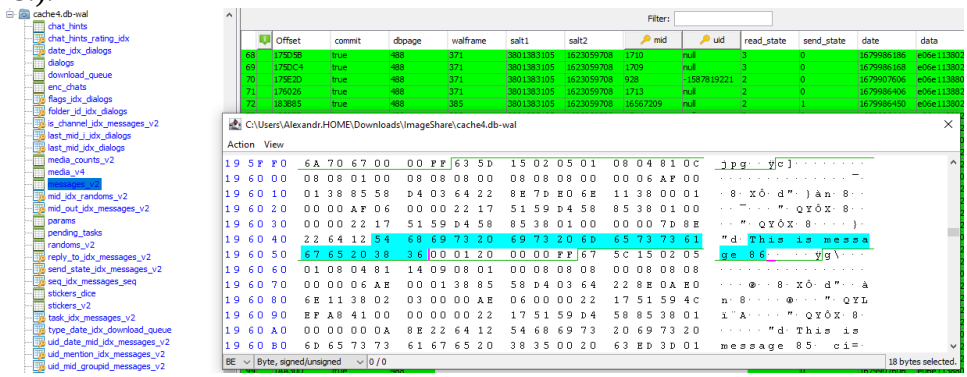


Рис. 7. Сообщение, удаленное из простого чата

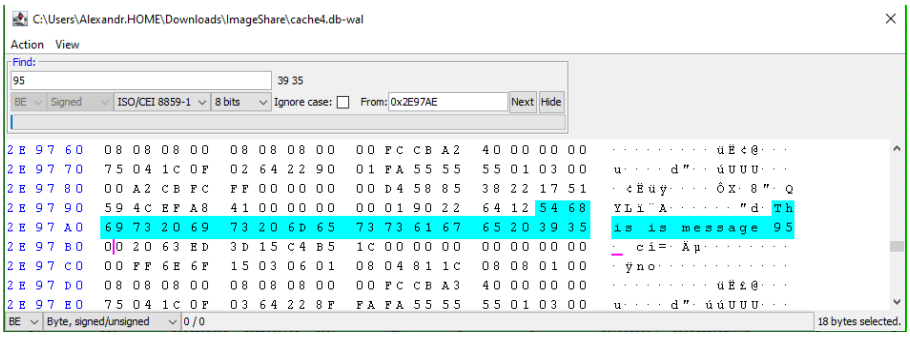


Рис. 8. Сообщение, удаленное из секретного чата

Из этого можем сделать вывод, что удаленные сообщения хранятся в файле cache4.db-wal.

Разберем, что означают некоторые цифры, которые хранятся в двоичном виде. 555555FA — флаг, обозначающий секретный чат, 41A8EF4C при переводе из шестнадцатеричной системы счисления дает 1101590348, это uid одного из пользователей в таблице Users (рис. 11.), 6422905D при переводе из шестнадцатеричной системы счисления дает 167998678, это время отправки сообщения. Покажем это на рис. 9.

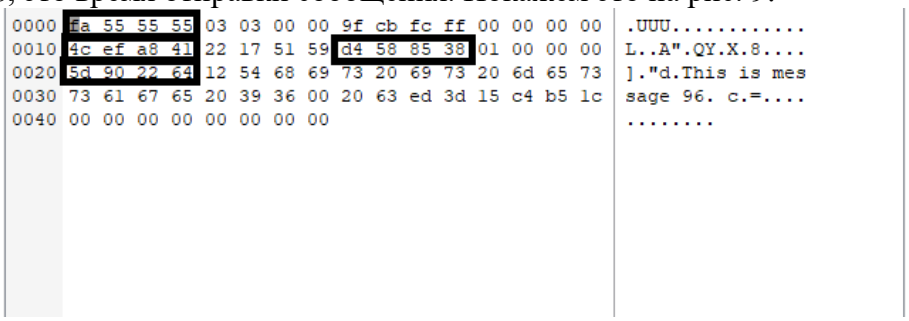


Рис. 9. Информация из сообщения секретного чата

38116EE0 — флаг, обозначающий простой чат, 41A8EF4C при переводе из шестнадцатеричной системы счисления дает 1101590348, это uid одного из пользователей в таблице Users (рис. 11.), 64228F12 при переводе из шестнадцатеричной системы счисления дает 1679986450, это время отправки сообщения. Покажем это на рис. 10.

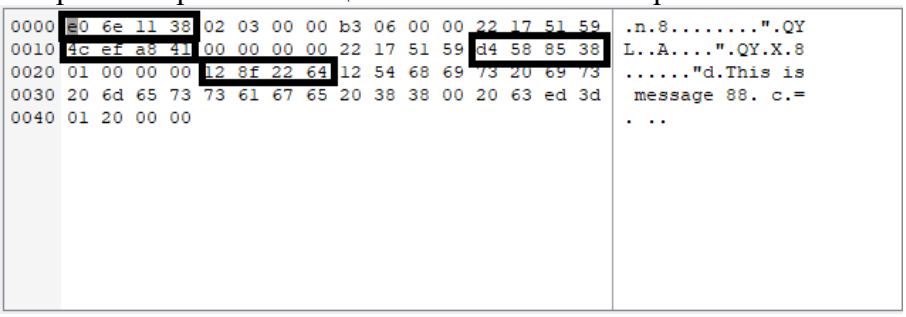


Рис. 10. Информация из сообщения обычного чата

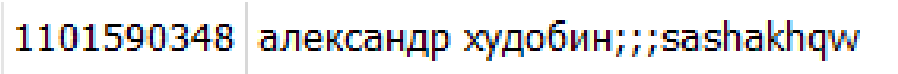


Рис. 11. Uid пользователя

Заключение

Результатом работы стало выявление и анализ следов, которые остаются после использования мессенджеров на персональных компьютерах на примерах популярных сервисов обмена мгновенными сообщениями WhatsApp и Viber. Произведено восстановление удаленных сообщений из мобильного приложения Telegram в простых и секретных чатах. Благодаря полученным данным был реализован мессенджер, который не требует от пользователя обязательной регистрации и который удаляет информацию, хранящуюся в базе данных.

Литература

1. Cosimo Anglano Forensic Analysis of Telegram Messenger on Android Smartphones / Cosimo Anglano, Cosimo Anglano, Marco Guazzone. – Computer Science Institute, Universit`a del Piemonte Orientale, Alessandria, 2017. – 50 с.

2. Alexandros Vasilaras Retrieving deleted records from Telegram / Alexandros Vasilaras, Donatos Dosis, Michael Kotsis, Panagiotis Rizomiliotis. – DFRWS 2022 APAC, 2022. – 15 с.

УДК 004.41

СОЗДАНИЕ ВИЗУАЛЬНЫХ ИНТЕРФЕЙСОВ ПРОГРАММНЫХ ПРОДУКТОВ С ПРИМЕНЕНИЕМ ДИНАМИЧЕСКИ ПОДКЛЮЧАЕМЫХ МОДУЛЕЙ

А. А. Худяков

Воронежский государственный университет

Введение

По мере развития информационных систем количество функций, возлагаемых на один программный продукт, сильно возросло. Большие монолитные приложения, сочетающие в себе не только богатые возможности для конечного пользователя, но и сложность поддержки и внедрения новых функций, сложность тестирования, долгое время сборки, запуска и прохождения автоматических тестов, были заменены на системы с применением сервис-ориентированной и микросервисной архитектур.

Предлагаемый в данной статье подход делает шаг вперед в сторону модульности уже в рамках конкретного сервиса. Он позволяет сделать подавляющую часть его кодовой базы полностью динамической, подгружая необходимые модули из хранилища данных в зависимости от конфигурации, что позволяет существенно менять функциональность сервиса без его перезапуска. В качестве примера такого подхода взято браузерное веб-приложение, однако при существовании соответствующих программных инструментов динамическая модульная архитектура может быть использована и в других программных продуктах, включая мобильные приложения и приложения для персональных компьютеров.

1. Подходы по использованию динамического контента

1.1. Динамический контент в сети

Подавляющая часть контента, которую можно увидеть в современных веб-приложениях, является полностью динамической. В социальных сетях, видео-хостингах, комментариях и отзывах к товарам и услугам контент создается самими пользователями приложения. В новостных сайтах и личных блогах этот контент создается авторами платформы, где нередко вместо обычного текста для хранения информации используют более продвинутые форматы и языки разметки, позволяющие задавать тексту структуру, включать в него изображения и настраивать стилизацию для отдельно взятых слов, предложений и блоков текста.

1.2. Продвинутые форматы хранения контента

Для хранения и создания статей в блогах и новостных сайтах, а также создания документации к информационным системам нередко используется Markdown – один из самых популярных и простых языков разметки, главным преимуществом которого является читабельность текста даже до преобразования в демонстрируемый пользователю вид.

Расширением и эволюцией Markdown является формат MDX [1], поддерживающий использование конструкций на языке JavaScript, импорт модулей с логикой и компонентов интерфейса, а также интеграцию вместе с популярными библиотеками для создания

пользовательских интерфейсов, такими как React, Preact, Vue, Solid и многими другими. Это позволяет использовать MDX не только для хранения и отображения статического контента, такого как статьи или файлы документации, но и для создания и хранения динамических компонентов со сложной логикой и открывает большие возможности для их стилизации.

Основным преимуществом MDX является тот факт, что он компилируется в JavaScript, а значит может быть динамически импортирован и подключен к уже существующему приложению без необходимости перекомпиляции и перезапуска сервера, на котором работает веб-приложение.

1.3. Веб-страница как контейнер для контентных блоков

На рис. 1 показано схематичное представление веб-страницы, состоящее из контентных блоков, каждый из которых при этом может являться статическим (всегда находиться на странице) или полностью динамическим (загружаться во время работы).

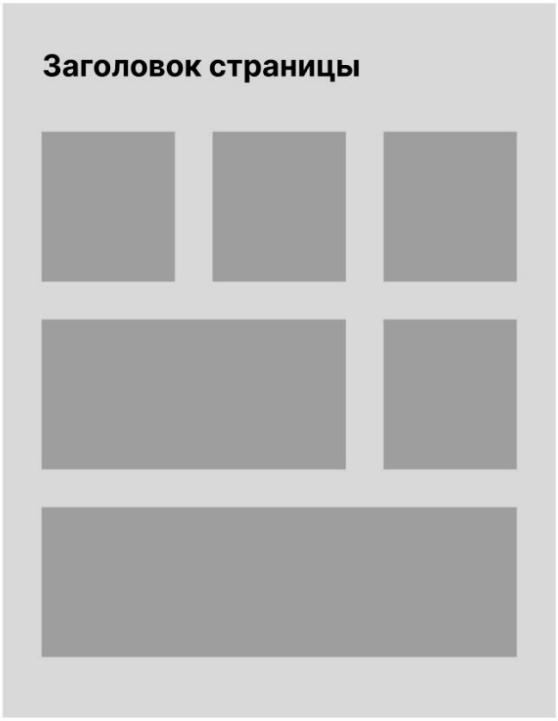


Рис. 1. Веб-страница с контентными блоками

Каждый блок может нуждаться в данных для отображения, которые могут как храниться в рамках самой системы, так и браться из сторонних источников. Для загрузки этих данных все блоки, как правило, будут использовать общую логику. Также все блоки на странице могут использовать общие элементы интерфейса, такие как кнопки, заголовки, выпадающие списки и любые другие повторно используемые компоненты, использовать общие модули-утилиты для форматирования данных, работы с датой и временем и многие другие.

В таком случае веб-страница не просто является агрегатором блоков и контролирует свою структуру, но и совершает инъекцию всех необходимых зависимостей каждому динамическому модулю и становится контейнером внедрения зависимостей. Структура страницы также может быть полностью динамической, редактируемой и храниться в базе

данных соответствующего сервиса.

Каждый блок может по умолчанию загружаться в специальном контейнере – окружении, ошибки из которого не распространяются за его пределы. Благодаря слабой связности блоков, ошибка времени выполнения или ошибка загрузки одного из них не приведет к остановке приложения. Она может быть корректно обработана и показана конечному пользователю в удобном формате.

2. Архитектура системы с динамически подключаемыми модулями

На рис. 2 в упрощенном виде показана система с микросервисной [2] архитектурой, в которой используется несколько сервисов бизнес-логики и сервис клиентского веб-приложения, использующий механизм динамических модулей. Команда разработки системы использует специальный сервер обработки подключаемых к веб-приложению модулей, который также позволяет проверять работоспособность каждого модуля путем запуска интеграционных и модульных тестов благодаря наличию очереди CI/CD.

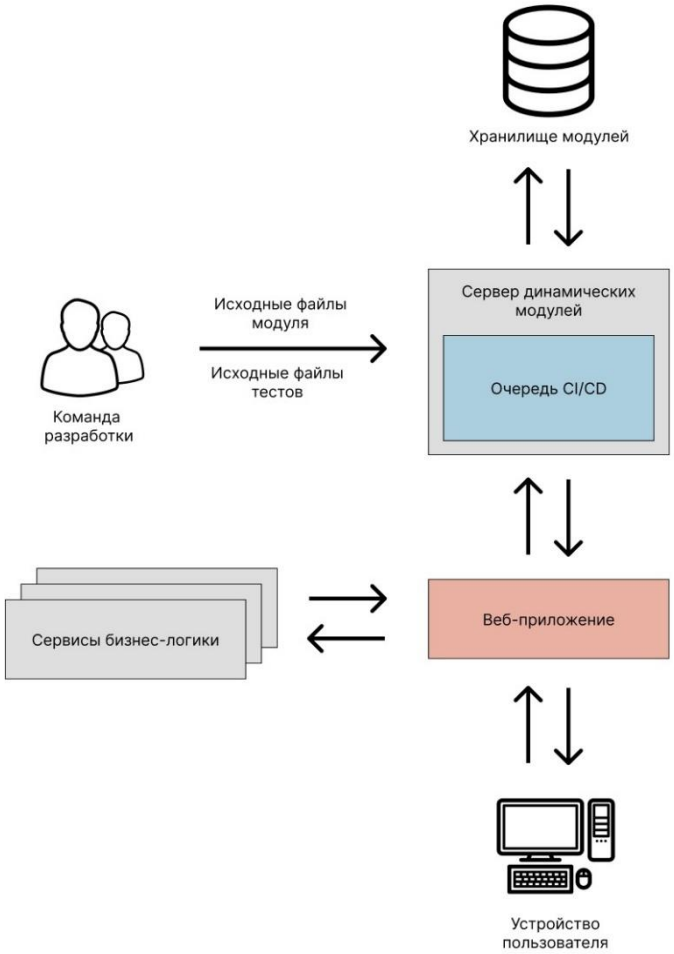


Рис. 2. Архитектура системы с динамическими модулями

Хранилище модулей также может хранить историю их версий, интегрируясь с системой контроля версий git, для возможности отката или использования разных версий для разных пользователей в рамках проведения A/B тестирования.

Заключение

Предложенный в данной статье подход к декомпозиции отдельных сервисов на динамически подключаемые модули открывает перспективу для создания приложений нового уровня сложности. Он позволяет не только производить гибкую настройку системы во время её работы, но и эффективно тестировать модули в отрыве от основного приложения, что особенно может быть полезно при существовании общей библиотеки компонентов логики и интерфейса, которая используется сразу несколькими программными продуктами. Это делает обновление всех приложений на новую версию библиотеки быстрым и простым, а уменьшение связности между компонентами одного сервиса и возможность проведения интеграционных тестов уменьшает потенциальное количество ошибок.

Литература

1. MDX. Официальная документация / Open Collective. – URL: https://mdxjs.com/docs (дата обращения 05.04.2023).
2. Boner J. Reactive Microservices Architecture / J. Boner. – Себастопол : O`Reilly Media Inc, 2016. – 54 с.

УДК 517.929.2

ПРОСТРАНСТВА СУММИРУЕМЫХ ФУНКЦИЙ С НОСИТЕЛЕМ НА ГРАФЕ

А. Чаговец

Воронежский государственный университет

Введение

В данной работе рассматриваются пространства суммируемых функций с носителем на графе. При исследовании краевых задач на графе основополагающим (центральным) вопросом является представление дифференциальных или иных соотношений в узловых местах графа (местах сочленения ребер графа), так как во внутренних точках ребер соотношения определяются классическими постановками, т. е. формализмами дифференциальных уравнений. Однако, вопрос разрешимости такого типа краевых задач требует еще и замены классических постановок так называемыми обобщенными (последнее десятилетие в специальной литературе используется другой термин – слабая постановка краевой задачи), для анализа которых необходимо формировать специальные пространства соболевского типа, элементы которых в определенном смысле удовлетворяют упомянутым выше соотношениям в узловых местах графа. Очевидное преимущество рассмотрения краевых задач в таких пространствах состоит в том, что для математического описания реальных процессов, т. е. для построения соответствующих математических моделей, класс непрерывных функций слишком узок (зачатую в таком классе решение задачи не существует), требуется расширения класса и использования суммируемых функций. Последние, как показали исследования прикладного характера в области оптимального управления, в высшей степени адекватности описывают свойства реальных процессов.

1. Определения и понятия

Везде ниже используются обозначения, понятия, определения и утверждения принятые в монографии [1, с. 88]. Используются следующие обозначения, присущие сети (см. рис. 1):

Γ -- пространственная сеть, математическая модель которой определена формализмами геометрического графа, каждому ребру γ графа (а значит, сети Γ) соответствует отрезок $[0,1]$;

Γ_0 -- пространственная сеть, каждому ребру γ которой соответствует интервал $(0,1)$ ($\overline{\Gamma_0} = \Gamma$);

$\partial\Gamma$ -- совокупность узлов, к которым присоединено только одно ребро (совокупность граничных узлов сети Γ);

$J(\Gamma)$ -- совокупность остальных узлов сети Γ (совокупность внутренних узлов сети Γ);

при некоторой фиксированной нумерации ребер сети через Z обозначено множество индексов узлов $\partial\Gamma$, через I -- число узлов $J(\Gamma)$: $J(\Gamma) = \{\xi_1, \xi_2, ..., \xi_I\}$.

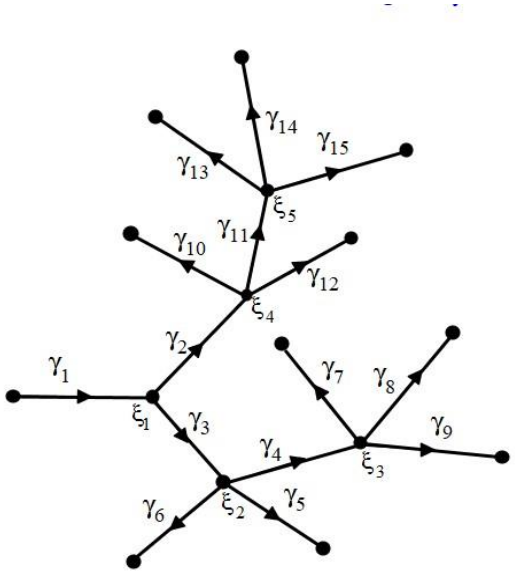


Рис. 1

2. Необходимость введения пространств

На сети Γ_0 (т. е. при $x \in \Gamma_0$, здесь и везде ниже x – пространственная переменная всех используемых в данной работе функций) рассматривается дифференциально-разностная система уравнений

$$\frac{1}{\tau}(y(k)-y(k-1))-\frac{d}{dx}\left(a(x)\frac{dy(k)}{dx}\right)+b(x)y(k)=f(k),\quad k=1,2,...,M, \qquad (7)$$

где $y(k):=y(x;k)$ и $f(k):=f(x;k)$, $k=1,2,...,M$. Отметим, что совокупность соотношений (1) есть полу-дискретизация эволюционного дифференциального уравнения вида

$$\Re y \equiv \frac{\partial y(x,t)}{\partial t}-\frac{\partial}{\partial x}\left(a(x)\frac{\partial y(x,t)}{\partial x}\right)+b(x)y(x,t)=f(x,t) \qquad (1^*)$$

по переменной $t \in (0,1)$ по прицину двухслойной схемы для t, причем $\tau=\frac{T}{M}$.

Для анализа системы (1) формируется пространства функций $y(k)$ ($k=1,2,...,M$), в дальнейшем такие пространства будут называться пространствами состояний уравнения (1).

Пространства состояний системы (1). Для формирования пространств состояний используются следующие классические пространства:

$L_p(\Gamma)$ ($p=1,2$) – лебегово пространство измеримых на Γ_0 функции $u(x)$, норма определена соотношением $\|u\|=(\int_\Gamma u^p(x)dx)^{1/p}$;

$W_2^1(\Gamma)$ -- соболевское пространство функций $y(x) \in L_2(\Gamma)$, для которых $\frac{dy}{dx} \in L_2(\Gamma)$, норма определена соотношением $\| u \|^1 = (\int_\Gamma (u(x)^2 + \frac{du(x)}{dx}^2) dx)^{1/2}$.

Введем симметричную форму

$$\ell(\mu,\nu) = \int_\Gamma \left(a(x) \frac{d\mu(x)}{dx} \frac{d\nu(x)}{dx} + b(x)\mu(x)\nu(x) \right) dx, \quad \mu,\, \nu \in W_2^1(\Gamma) \tag{8}$$

непрерывную по каждому, отдельно взятому μ или ν, где $a(x),\, b(x) \in L_2(\Gamma)$ и

$$0 < a_* \le a(x) \le a^*,\ |b(x)| \le \beta,\ x \in \Gamma_0,.$$

$a_* = const,\ a^* = const,\ \beta = const$ заданы. Несложно показать, что множество $\Omega_a(\Gamma)$ непрерывных на Γ функций $u(x)$, для которых

$$\sum_{\gamma \in R(\xi)} a(1)_\gamma \frac{du(1)_\gamma}{dx} = \sum_{\gamma \in r(\xi)} a(0)_\gamma \frac{du(0)_\gamma}{dx} \quad \forall \xi \in J(\Gamma),$$

где $R(\xi)$ и $r(\xi)$ – наборы ребер, ориентированных к ξ и от ξ, соответственно, принадлежит $W_2^1(\Gamma)$.

Определение 1. *Замыкание множества $\Omega_a(\Gamma)$ в норме $W_2^1(\Gamma)$ задает соболевское пространство $W^1(a;\Gamma)$.*

Определение 2. *Если считать, что элементы $\Omega_a(\Gamma)$, таковы, что $u(x)|_{\partial\Gamma} = 0$, тогда замыкание такого множества в норме $W_2^1(\Gamma)$ задает соболевское пространство $W_0^1(a;\Gamma)$.*

Замечание 1. *Ясно, что $W_0^1(a;\Gamma) \subset W^1(a;\Gamma)$. Пространство $W_0^1(a;\Gamma)$ используется для анализа дифференциально-разностной системы* (1), *когда функции $y(k)$ ($k = 1,2,...,M$) в граничных узлах $\partial\Gamma$ удовлетворяют условиям Дирихле, пространство $W^1(a;\Gamma)$ используется для иных более сложных условий.*

3. Использование пространств для анализа систем (1) и (1*)

Покажем на примере пространства $W_0^1(a;\Gamma)$, что означает

$$y(0) = \varphi(x), \quad y(k)|_{x \in \partial\Gamma} = 0,\ k = 1,2,...,M. \tag{9}$$

Сделаем следующие предположения: $f(k) \in L_2(\Gamma)$, $k = 1,2,...,M$.

Определение 3. *Удовлетворяющий интегральным тождествам*

$$\int_\Gamma y(k)_t \eta(x)dx + \ell(y(k),\eta) = \int_\Gamma f(k)\eta(x)dx,\ \ k = 1,2,...,M,\quad \forall \eta(x) \in W_0^1(a,\Gamma),$$
$$y(0) = \varphi(x),$$

набор функций $y(k) \in W_0^1(a;\Gamma)$, $k = 1,2,...,M$, *является слабым решением дифференциально-разностной системы* (1), (3); *здесь* (*см,* (2)),

$$y(k)_t = \frac{1}{\tau}[y(k) - y(k-1)],$$
$$\ell(y(k),\eta) = \int_\Gamma \left(a(x)\frac{dy(k)(x)}{dx}\frac{d\eta(x)}{dx} + by(k)(x)\eta(x) \right) dx.$$

Таким образом, исходя из приведенного выше эволюционного уравнения переноса (1*) (т. е. уравнения переноса при непрерывно изменяющимся времени), можно дифференциально-разностную систему (1), (3) интерпретировать как перенос при дискретно изменяющемся времени [2].

Замечание 2. *Уравнение* (1) *порождает краевую задачу* (1), (3) *относительно неизвестной функции* $y(k) \in W_0^1(a;\Gamma)$ *при фиксированном* $k = 1,2,...,M$ (*разрешимость аналогичных задач рассмотрена в* [2] *и* [3]). *Откуда следует существование слабого решения дифференциально-разностной системы* (1), (3) $y = \{y(k) \in W_0^1(a;\Gamma),\ k = 1,2,...,M\}$.

Замечание 3. *Разрешимость дифференциальной системы* (1*) *с заданными начальным и граничным условиями устанавливается построением последовательности приближений ее слабых решений по определенным функциям* $y(k) \in W_0^1(a;\Gamma)$, $k = 1,2,...,M$, *и последующего перехода к пределу при* $M \to \infty$.

Заключение

Рассмотренное использование пространств для анализа дифференциально-разностных систем позволяет интерпретировать перенос при дискретно изменяющемся времени и получить условие слабой разрешимости дифференциально-разностных систем с пространственными переменными. Полученные результаты могут быть применены для анализа динамики процессов в различных областях, таких как физика, химия и биология, а также в задачах оптимизации, возникающих при моделировании сетеподобных процессов переноса сплошных сред.

Научный руководитель доцент, доктор физ.-мат.наук, профессор кафедры уравнений в частных производных и теории вероятностей ВГУ, Провоторов Вячеслав Васильевич.

Литература

1. Провоторов В. В., Волкова А. С. *Начально-краевые задачи с распределенными параметрами на графе*. Воронеж: Научная книга. 2014. 188 с.

2. Zhabko A. P., Provotorov V. V., Shindyapin A. I. Optimal control of a differential-difference parabolic system with distributed parameters on the graph // Vestnik of Saint Petersburg

University. Applied Mathematics. Computer Science. Control Processes. 2021. vol.~17. iss.~4. pp.~433--448.
https://doi.org/10.21638/11701/spbu10.2021.411

3. Podvalny S L, Provotorov V V and Podvalny E S 2017 The controllability of parabolic systems with delay and distributed parameters on the graph // Procedia Computer Sciense 12th. Cep. "12th. International Symposium Intelligent Systems, INTELS 2016" 2017. pp. 324-330.

УДК 519.15

АЛГОРИТМ РАСТЯЖЕНИЯ НИТИ ДЛЯ ПОСТРОЕНИЯ МАРШРУТОВ

У. В. Чеботарев

[1]*Воронежский государственный университет*

Аннотация. В статье предложен эвристический алгоритм для построения маршрута на прямоугольном поле с препятствиями. Идея алгоритма опирается на создание траектории движения мобильного робота путём растягивания и искривления траектории, проведённой от некоторой начальной позиции до целевой. Алгоритм строит ориентированный путь в двумерном пространстве.

Ключевые слова: схема «разделяй и властвуй», жадный алгоритм, граф

Введение

Задача планирования маршрутов – одна из самых известных задач, которые имеют как самостоятельное значение (навигация, оптимизация линий коммуникаций, определение надежности систем), так и применяются к конструированию некоторых алгоритмов (например, лучевой алгоритм, алгоритм левой руки, алгоритм поворотов Креша (Crash), алгоритм A*, алгоритм Дейкстры). Широко используется данная задача в игровых приложениях. Известны различные модификации и обобщения данной задачи (планирование маршрутов в динамически изменяющейся среде, в условиях частичной неопределенности). К настоящему времени существует несколько базовых алгоритмов для решения данной задачи. В [9] предложен алгоритм, основанный на разбиении плоской карты на ячейки, которые далее представляются вершинами неориентированного графа, и уже в этом графе применяются специальные графовые алгоритмы поиска маршрута. В [7] предложен нейросетевой алгоритм поиска маршрута. Поскольку задача планирования маршрута может быть сформулирована как оптимизационная, то известны подходы, в основе которых лежат эвристические алгоритмы для нахождения субоптимальных решений, например, алгоритм муравьиных колоний [8].

1. Задача планирования маршрутов

1.1. Постановка задачи

Пусть имеется прямоугольное поле, по которому может перемещаться мобильный объект, способный двигаться в произвольном направлении. Размеры мобильного объекта строго заданы, для его движения необходимо незанятое место не менее чем в два раза превосходящее диаметр самого мобильного объекта. Для удобства можно принять за форму объекта круг с заданным радиусом. Предположим, что на поле имеются выпуклые статические препятствия, причем, если расстояние между ними менее двух диаметров мобильного робота, то такие препятствия можно рассматривать в совокупности как одно, но требование на выпуклость препятствий обязано сохраняться. В связи с этим, оказывается, что перемещение для объекта в окрестности препятствий не является возможным. Препятствия являются статическими и не могут изменить никаких своих характеристик. Для мобильного объекта задаются две обязательно свободные позиции – исходная, в которой он находится в начальной момент времени, и целевая – та, в которую объект должен переместиться. Позиция может

задаваться как точка с окрестностью равной двойному размеру мобильного робота.

Введем для поля с препятствиями систему координат, ориентация которой совпадает с вертикальной и горизонтальной границами поля, и проведём прямую, соединяющую начальную и целевую точки. Необходимо построить маршрут для движения объекта из начального положения в целевое с учетом имеющихся препятствий. За маршрут будет приниматься последовательность точек с расстоянием меньше двойного диаметра объекта.

1.2. Формализация задачи

Определим Декартову систему координат для прямоугольного поля, где нижний левый угол совпадает с точкой (0;0).

Введем некоторые понятия и обозначения:

L – отрезок, соединяющий начальную и конечную точки позиций мобильного робота;

δ – диаметр объекта;

$P = \{p_1, ..., p_r\}$ – множество препятствий, каждое p^k может задаваться функционально или системой неравенств (чтобы размеры мобильного объекта можно было не учитывать и принять его как материальную точку, далее все препятствия будут рассматриваться вместе с собственной δ-окрестностью);

T_i – i -я точка, через которую проходит маршрут.

Мобильный объект может перемещаться в любом направлении, но не может находиться в δ-окрестности препятствия. Исходная позиция объекта в ячейке s, конечная – в ячейке t. Решение задачи ищется в форме упорядоченной последовательности координат точек T_i, расстояние между которыми 2δ, соединив которые, получим ломаную маршрута, перемещаясь по которому, мобильный объект прибудет в целевую точку t, начиная свое движение из точки s.

2. Решение поставленной задачи

2.1. Основные предположения

Моделью для решения данной задачи может служить схема «разделяй и властвуй», где за эталонный маршрут будет браться отрезок L, который будет разбиваться на меньшие отрезки как отдельные подзадачи, которые будут изменяться, если они пересекают препятствие. Решение удобно представлять графически.

Выделим следующие возможные ситуации, которые могут возникнуть при движении мобильного объекта во время нахождения кратчайшего пути и привести к ошибке выполнения:

– путь из s в t не существует, соответственно, нет и возможности построить маршрут, тогда необходимо завершить алгоритм, это можно выяснить, если невозможно разбить две точки маршрута так, что между ними ни окажется препятствия;

– расстояние между точками стало меньше 2δ, тогда очевидно, что препятствий на отрезке между ними нет, и более уменьшать рассматриваемую область не имеет смысла;

– начальная позиция объекта и пункт назначения могут совпадать, тогда перемещение объекта будет равно нулю.

Основные предположения:

1) все точки, принадлежащие маршруту, не должны находиться на препятствии;
2) маршрут содержит конечное количество точек.

2.2. Описание алгоритма

Идея алгоритма заключается в проверке каждой из точек маршрута (предположение 1) и заменой такой точки, для которой (предположение 1) не выполнено до тех пор, пока это не станет справедливо для всех точек (рис. 1).

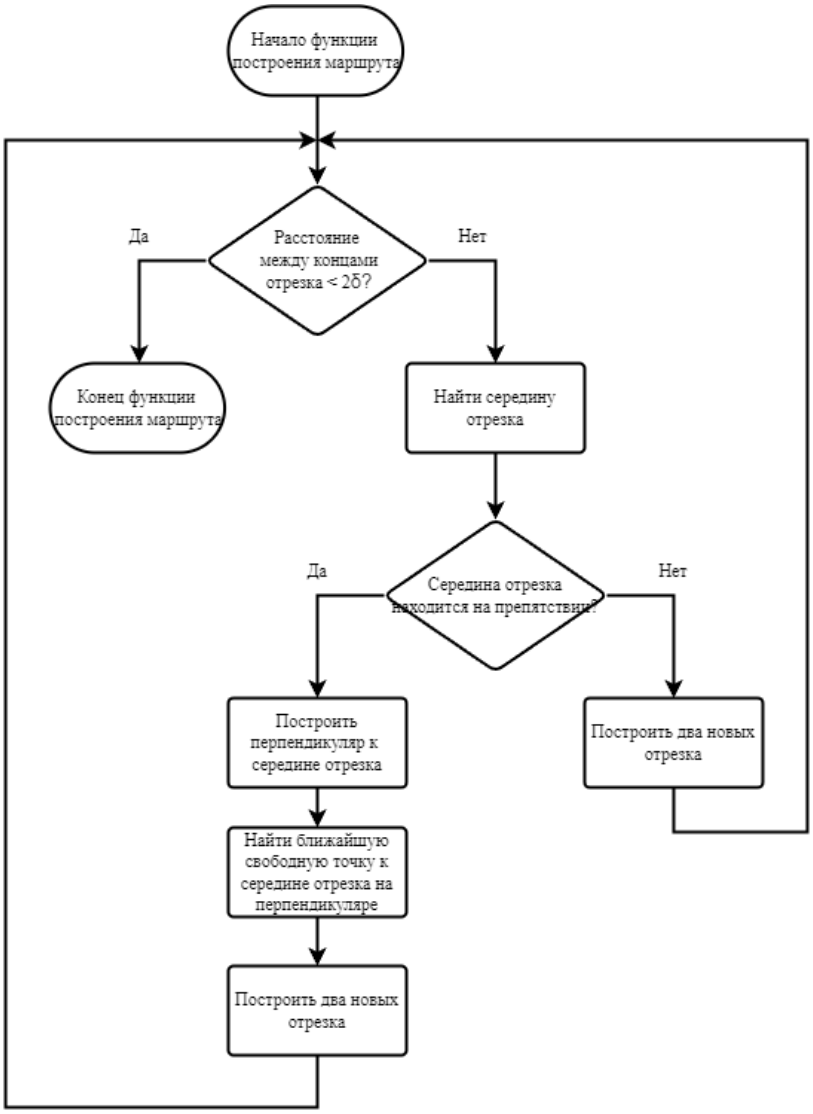


Рис. 1.

Очевидно, что оптимальным маршрутом из s в t будет отрезок L. Пусть точка A_1 – начало отрезка L, B_1 – конец, а O_1 – середина A_1B_1. Проверим, пересекает ли точка O_1 препятствие, если нет, рассмотрим два новых отрезка A_1O_1 и O_1B_1, иначе проведём перпендикуляр к данному отрезу через точку O_1, и с шагом δ найдём на нём такую точку O_1^*, что она удовлетворяет (предположение 1), а расстояние между O_1 и O_1^* минимально, рассмотрим два новых отрезка $A_1O_1^*$ и $O_1^*B_1$, и так далее рекурсивно, подставляя новые

границы отрезков, до тех пор пока расстояние между новыми точками не станет меньше 2δ (рис. 2).

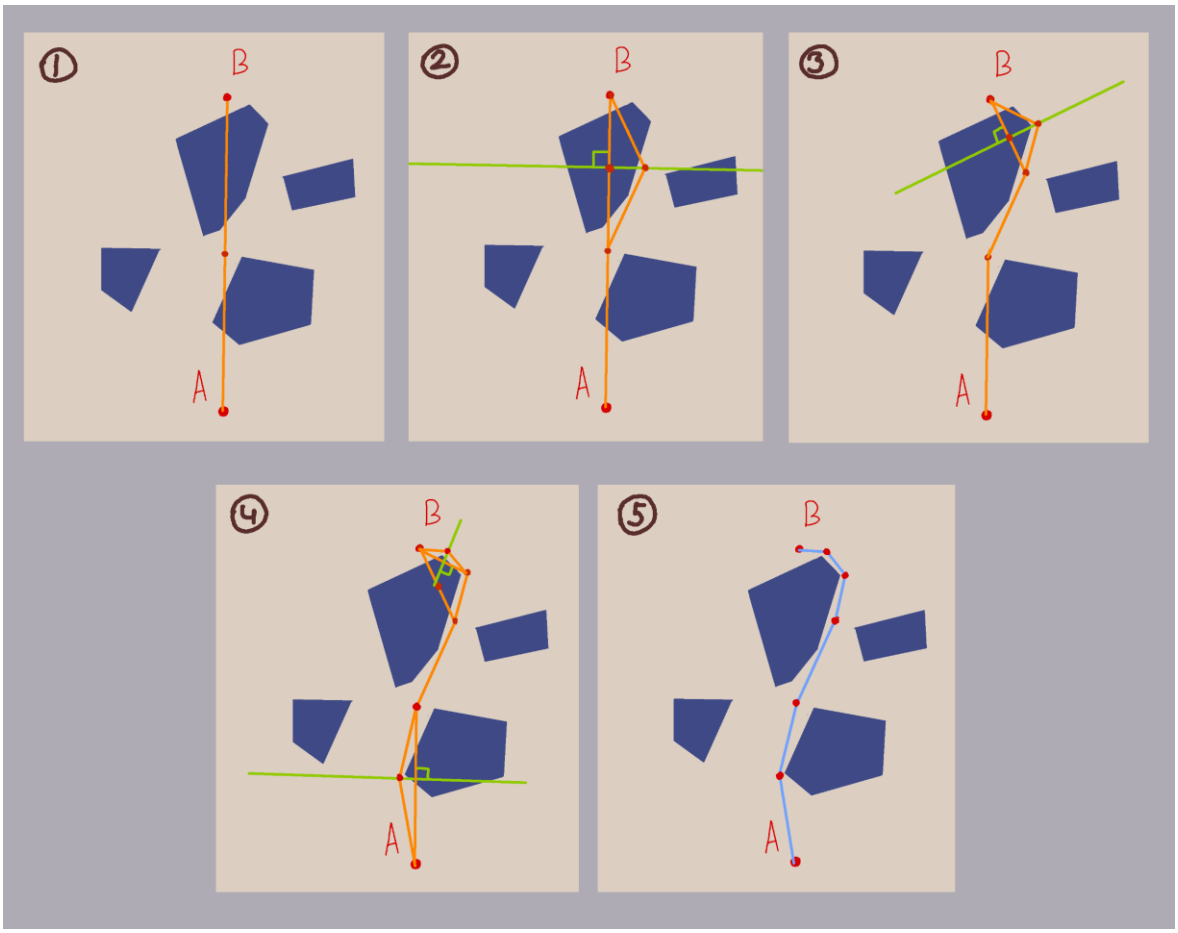


Рис. 2.

Заключение

Предложенный рекуррентный алгоритм интересен тем, что в отличие от многих при решении не сводится к графам, а также не использует классических алгоритмов разбиения плоскости на ячейки и нахождения маршрутов или кратчайшего пути, он применим в прикладных задачах (например, создание дорожного полотна, обход препятствий на пересеченной местности), не требует адаптации и полного обхода поля, а в некоторых случаях, при уменьшении количества и размера препятствий, построенные им маршруты приближаются к кратчайшему пути.

Для предложенного алгоритма возможны дальнейшие модификации (например, нахождения маршрута с вогнутыми или динамическими препятствиями, добавление территорий с разными условиями проходимости).

Литература

1. Кристофидес, Н. Теория графов. Алгоритмический подход / Н. Кристофидес. – Москва : Мир, 1978. – 427 с.

2. Леденева, Т.М. Некоторые алгоритмы прикладной теории графов / Т.М. Леденева. –

Воронеж : Издательский дом ВГУ, 2021. – 35 с.

3. Большаков А.А. Планирование траектории движения мобильного робота / А.А. Большаков, М.Ф. Степанов, А.М. Степанов, А.Ю. Ульянина // Вестник Саратовского государственного технического университета, 2010. – №4(51) – Вып.3. – С.176-180.

4. Кристофидес Н. Теория графов. Алгоритмический подход. / Н. Кристофидес Пер. с англ. – М.Ж Мир, 1988. – С

5. Подураев Ю.В., Актуальные проблемы мехатроники / Ю.В.Подураев // Мехатроника. Автоматизация. Управление, – 2007. – №4. – С. 50-53

6. Desai J.P., Ostrowski J.P., Kumar V. Modeling and control of formations of nonholonomic robots. IEEE Transactions on Robotics and Automation, 17(6):905-908, 2001

7. Юдинцев Б.С. Оптимизация методов планирования траекторий групп мобильных роботов с использовани ем нейронной карты // Мавлютовские чтения (Т. 3) – Уфа, 2011. – С. 188–189.

8. Шаоцзянь, Сун. Исследовательский шлюз по оптимизации сетей эхосостояний на основе алгоритма муравьиной колонии / Сун Шаоцзянь, Ван Яо, Линь Сяофэн // Вычислительная техника и наука. – 2017. – С. 2326-2332.

9. Мигранов А.Б., Юдинцев Б.С. Даринцев О.В. Ультразвуковая сенсорная система для реализации интеллектуального управления движением группы мобильных роботов // Труды института механики (Вып. 7). – Уфа, Нефтегазовое дело, 2010. – С. 109–117

УДК *519.688*

О ВЫБОРЕ АЛГОРИТМОВ ГЕНЕРАЦИИ НАВИГАЦИОННЫХ СЕТОК ДЛЯ ИСПОЛЬЗОВАНИЯ В ИЗМЕНЯЮЩЕМСЯ ИГРОВОМ ОКРУЖЕНИИ

Н.М. Чернышов

Воронежский государственный университет

Введение

Одной из известных математических задач является задача поиска кратчайшего пути между двумя точками. Существуют различные подходы к решению этой задачи [1]. Одним из таких подходов является применение навигационной сетки – структуры данных, состоящей из множества выпуклых многоугольников, которые определяют область, в которой объект может свободно перемещаться. Поиск пути внутри одного из этих многоугольников может быть выполнен тривиально по прямой линии, потому что многоугольник выпуклый и проходимый, а поиск пути между многоугольниками в сетке может быть выполнен с помощью одного из большого количества алгоритмов поиска в графе [2]. Такой подход позволяет избежать дорогостоящих с точки зрения вычислений проверок обнаружения столкновений с препятствиями. Этот подход используется как в робототехнике для обхода роботом препятствий, так и в компьютерных играх для перемещения игровых персонажей среди объектов окружения.

В некоторых компьютерных играх игровое окружение может изменяться с течением времени. Например, могут произойти такие изменения, как строительство нового здания, падение метеорита или осушение водного канала. Такие изменения требуют преобразований формы или построения новой навигационной сетки, что может потребовать значительных вычислительных мощностей.

В работе [3] был реализован алгоритм построения нерегулярной четырехугольной сетки, а также приведен пример применения полученной сетки для создания игрового окружения. Особенностью данной сетки является независимое хранение данных о каждом шестиугольнике, содержащем четырехугольную сетку. На основе нерегулярной сетки, полученной в каждом шестиугольнике, можно осуществить построение навигационной сетки. Такую сетку будет легко преобразовывать при изменении игрового уровня (необходимо перестроить сетку только в тех шестиугольниках, где произошли изменения), а построение новых фрагментов навигационной сетки можно выполнять параллельно.

В статье представлены описания некоторых алгоритмов и известные результаты сравнения этих алгоритмов как между собой, так и с другими алгоритмами генерации навигационных сеток с целью поиска минимального по времени выполнения и сложности сетки алгоритма генерации навигационной сетки для изменяющегося в реальном времени игрового пространства.

В дальнейшем предполагается реализовать алгоритм генерации навигационных сеток на основе результатов, полученных в работе [3], а также сравнить сложность создаваемой сетки и время выполнения алгоритма с результатами применения реализаций других алгоритмов генерации навигационных сеток.

1. Рассмотренные алгоритмы генерации навигационных сеток

Ниже представлены алгоритмы генерации навигационных сеток.

1. Равномерная прямоугольная сетка.
2. Триангуляция локальных зазоров.
3. Карта явных коридоров.
4. Алгоритм Recast.
5. Автоматический генератор навигационных сеток ANavMG.
6. Генератор почти оптимальных навигационных сеток NEOGEN.
7. Навигационный граф на многослойной и неровной местности.
8. Алгоритм генерации навигационных сеток, представленный в Unreal Engine 3.

Агент навигации — объект, осуществляющий перемещение по навигационной сетке.

Наиболее простым алгоритмом для создания навигационных сеток является алгоритм построения равномерной прямоугольной сетки. Каждая проходимая область на такой сетке представляет собой прямоугольник. Построение сетки осуществляется при помощи применения одного из алгоритмов вокселизации к объектам окружения, являющимся препятствиями. Возможно также отсечение областей, которые являются недосягаемыми для агентов.

Триангуляция локальных зазоров [4] вычисляется с помощью операций уточнения триангуляции Делоне с ограничениями [5] входного набора двумерных препятствий. Уточнения предназначены для обеспечения того, чтобы два значения локального зазора, хранящиеся в каждом ребре, были достаточными для точного определения того, может ли диск произвольного размера пройти через какие-либо узкие проходы сетки. Это свойство необходимо для правильного и эффективного извлечения путей с зазором непосредственно из триангуляции без необходимости нахождения срединной оси — геометрического объекта, представляющего собой геометрическое место точек плоскости, равноудаленных от границы фигуры.

Построение карты явных коридоров [6] осуществляется следующим образом. Сначала вычисляется срединная ось каждой двумерной пешеходной области в окружающей среде. Полученный набор срединных осей объединяется в единую многоуровневую срединную ось на основе связей в окружающей среде. При дополнении этой структуры дополнительными отрезками, соединяющими срединную ось с ближайшими препятствиями, получим разбиение многослойной среды на набор проходимых областей. Эти проходимые области определяют навигационную сетку таким образом, что все точки в проходимой области имеют одни и те же ближайшие препятствия.

Алгоритм генерации навигационных сеток Recast [7] получает на вход множество треугольников и выполняет их вокселизацию, производя при этом построение многослойного поля высоты. Воксели в областях, где агенты не могут перемещаться, удаляются при помощи фильтров. Проходимые области, образованные оставшимися вокселями, разбиваются на наборы двумерных полигональных областей при помощи применения алгоритма водораздела к полям высоты. Навигационные полигоны генерируются путем триангуляции и соединения соседних двумерных полигональных областей.

Автоматический генератор навигационных сеток ANavMG [8] создает близкую к оптимальной выпуклую декомпозицию игровой сцены в виде графа ячеек и порталов (cell-and-portal graph) – набора ячеек, являющихся выпуклыми областями в пространстве, и порталов, которые являются общими с выпуклыми областями ребрами, через которые может пройти агент. Алгоритм применим только для двумерных игровых пространств.

Генератор почти оптимальных навигационных сеток NEOGEN [9] расширяет возможности алгоритма ANavMG, позволяя строить многоуровневые навигационные сетки. Алгоритм сперва осуществляет вокселизацию геометрии сцены. Затем воксельная сетка

обрабатывается для выделения множества слоев, содержащих проходимые области игровой сцены. Каждый слой затем обрабатывается с целью выделения контуров двумерных проходимых многоугольников и препятствий, являющихся отверстиями в проходимых многоугольниках. Для каждого слоя выполняется построение графа ячеек и порталов при помощи алгоритма ANavMG, после чего графы объединяются для получения целостной навигационной сетки.

Алгоритм построения навигационного графа на многослойной и неровной местности [10] строит навигационные сетки как для двумерных, так и многослойных игровых окружений, так, чтобы вычисление пути для перемещения больших групп агентов занимало относительно мало времени для выполнения этой операции в реальном времени. Построение графа осуществляется следующим образом. Исходное окружение подразделяется на проходимое пространство и препятствия. Для проходимого пространства выполняется построение диаграммы Вороного, после чего на основе диаграммы строится множество не перекрывающих друг друга выпуклых ячеек. После этого строится граф смежности ячеек, который формирует навигационный граф.

Алгоритм построения навигационной сетки в среде разработки Unreal Engine 3 [11] состоит из 3 этапов. На первом этапе, начиная с каждой позиции, выбранной пользователем, игровое пространство заполняется при помощи алгоритма заливки для определения областей, достижимых навигационными агентами. На втором этапе происходит упрощение сетки при помощи слияния соседних квадратов в вогнутые плиты, разделенные только разницей в угле наклона. Вогнутые плиты затем разбиваются на выпуклые многоугольники. На последнем этапе осуществляются определение проходимых ребер между многоугольниками и создание сетки препятствий.

2. Известные результаты сравнения алгоритмов

В работе [12] представлено сравнение по множеству критериев алгоритмов триангуляции локальных зазоров, генерации карты явных коридоров, Recast, NEOGEN и навигационных графов на многослойной и неровной поверхности. Согласно результатам работы, алгоритм триангуляции локальных зазоров является самым эффективным по времени выполнения, однако в измерения не был включен этап предварительной обработки игровых окружений, необходимой для корректного функционирования алгоритма на трехмерных игровых сценах. При увеличении масштаба игровых окружений точные методы [4, 6] производят генерацию быстрее, чем методы, использующие воксельную сетку. Самым быстрым алгоритмом, использующим воксельную сетку, является NEOGEN.

В работе [13] представлены результаты сравнения по времени выполнения алгоритмов Recast и NEOGEN. Согласно результатам работы, алгоритм NEOGEN быстрее алгоритма Recast в большинстве случаев, однако для сравнения использовалось малое количество игровых окружений. Также стоит отметить, что при сравнении не учитывалась особенность алгоритма NEOGEN отсекать недостижимые навигационным агентом области игрового пространства, которые алгоритм Recast использовал при генерации конечной навигационной сетки. На рис. 1 показан пример навигационной сетки , построенной алгоритмами Recast и NEOGEN для некоторой игровой сцены. NEOGEN не выполнил построение навигационной сетки в верхней части сцены, поскольку она является недостижимой из нижней части, в то время как Recast выполнил построение навигационной сетки для всего игрового уровня. Эта особенность связана с тем, что NEOGEN использует алгоритм заливки для получения навигационных слоев.

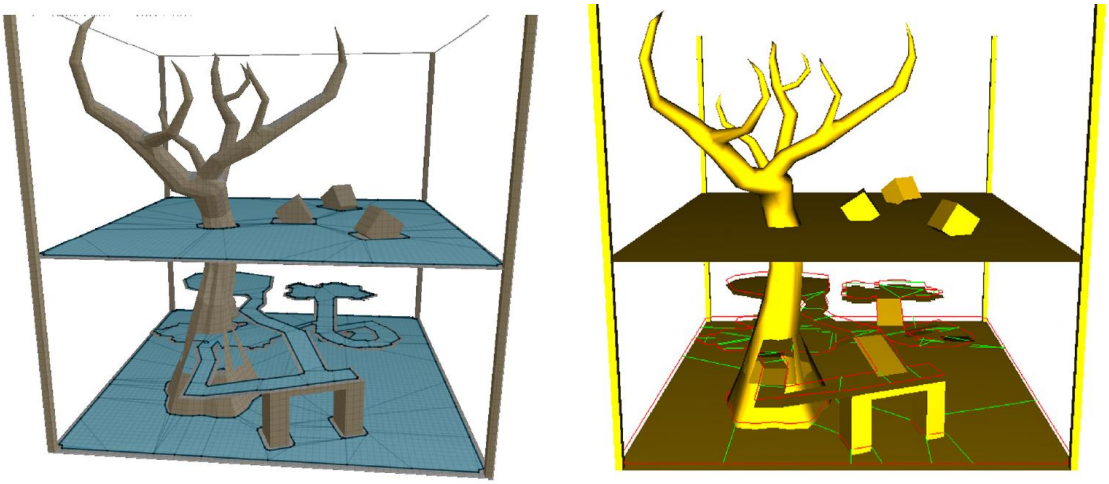


Рис. 1. Пример навигационной сетки, построенной алгоритмами Recast (слева) и NEOGEN (справа)

В работе [13] также указывается, что алгоритм генерации навигационных сеток, представленный в Unreal Engine 3, создает множество плохо обусловленных ячеек, что может приводить к случаям некорректного построения связей между соседними многоугольниками и приводить к возникновению артефактов при поиске пути навигационными агентами. Кроме того, отмечается, что разбиение на квадраты, полученное на неровных участках игровой сцены этим алгоритмом, чрезмерно сегментировано, что значительно увеличивает вычислительные затраты при поиске пути.

3. Выбор алгоритмов для реализации и сравнения

Для реализации и сравнения между собой и с алгоритмом генерации навигационной сетки из нерегулярной четырехугольной сетки [3] были выбраны следующие алгоритмы: равномерная прямоугольная сетка Recast, NEOGEN. Алгоритмы [4, 6] требуют проецирование трехмерных объектов на плоскость, что является крайне ресурсоемкой задачей для изменяющегося игрового окружения. Алгоритм построения навигационных графов [10], согласно результатам работы [12], медленнее алгоритма Recast. Чтобы выполнить корректное сравнение алгоритмов NEOGEN и Recast, необходимо модифицировать NEOGEN таким образом, чтобы алгоритм генерировал навигационную сетку не только в досягаемых из выбранной пользователем точки областях игрового пространства, но и в остальных областях.

4. Описание реализуемых алгоритмов

4.1. Алгоритм генерации равномерной прямоугольной навигационной сетки

Алгоритм получает на вход множество трехмерных моделей объектов игровой сцены и набор параметров a, b, ε и возвращает равномерную прямоугольную сетку, которую можно использовать для навигации агентов. Алгоритм состоит из двух шагов:

1. Вокселизация полигональных сеток объектов игровой сцены.
2. Удаление связей вокселей, между которыми невозможно осуществить переход.

Для вокселизации пространства можно воспользоваться алгоритмом растеризации [14]. Результатом растеризации является равномерная прямоугольная сетка, содержащая в каждой ячейке наибольшую среди всех треугольников исходной геометрии высоту. Воспользуемся значениями высоты, хранящимися в растровой сетке, для построения равномерной прямоугольной сетки, содержащей в каждой ячейке логическое значение проходимости данной ячейки. Для этого рассмотрим все пары соседних в окрестности фон Неймана ячеек и установим непроходимыми те пары, которые образуют угол наклона с горизонтальной плоскостью больший, чем некоторое заданное значение ε. Также установим непроходимыми те ячейки, высота которых не лежит на отрезке $[a, b]$.

4.2. Алгоритм Recast

Алгоритм получает на вход множество трехмерных моделей объектов игровой сцены. Для реализации алгоритма необходимо выполнить следующие шаги [15]:

1. Вокселизация трехмерных моделей.
2. Построение навигационного пространства из вокселей.
3. Разбиение навигационного пространства на регионы при помощи алгоритма водораздела.
4. Трассировка и упрощение контуров регионов.
5. Триангуляция контуров регионов и соединение соседних треугольников для получения навигационной треугольной сетки.

Вокселизацию объектов игровой сцены можно выполнить при помощи алгоритма растеризации. Навигационное пространство строится из вокселей, которые не слишком низко и не слишком высоко. В навигационное пространство не включаются воксели, образующие слишком крутые склоны с горизонтальной плоскостью.

Для выполнения разбиения пространства при помощи алгоритма водораздела необходимо предварительно выполнить дистанционное преобразование навигационного пространства, где для каждого проходимого вокселя хранится расстояние до ближайшего препятствия. Построить карту дистанционных преобразований можно при помощи алгоритма Мейстера-Рурдинка-Хесселинка за линейное время [16]. Разбить пространство на области можно при помощи алгоритма водораздела Винсенса-Сойля [17], использованного в оригинальной реализации Recast.

Трассировка контуров осуществляется следующим образом: для каждого региона, полученного в результате сегментации воксельного пространства, выполняется выбор начальной точки. После этого, начиная с начальных точек, выполняется обход всех соседних вокселей с идентификатором региона, совпадающим с идентификатором начальной точки. Воксели, граничащие с другими регионами, добавляются в множество угловых точек с сохранением идентификаторов соседних регионов для последующего связывания регионов между собой.

В зависимости от величины шага вокселизации контур может иметь различное количество точек. Если контур будет иметь большое количество углов, то количество треугольников, необходимое для покрытия формы контура, будет также достаточно велико. Упростить контуры можно при помощи алгоритма Рамера-Дугласа-Пекера [18].

Для триангуляции упрощенных контуров можно использовать метод отсечения ушей [19], поскольку регионы, образованные алгоритмом водораздела, не содержат дыр и каждому региону соответствует всего один контур.

4.3. Модифицированный алгоритм NEOGEN

Генератор почти оптимальных сеток для трехмерных многослойных сред NEOGEN [9] получает на вход множество многоугольников, образующих многослойную трехмерную среду. Построение навигационной сетки при использовании исходного алгоритма осуществляется следующим образом:

1. Вокселизация исходных многоугольников.
2. Определение из воксельного пространства и маркировка навигационных слоев.
3. Обработка слоев.
4. Генерация навигационной сетки при помощи алгоритма ANavMG для каждого слоя.
5. Объединение сеток навигационных слоев.

Изменим алгоритм так, чтобы результирующая навигационная сетка включала в себя все возможные области, в которых может находиться навигационный агент. Для вокселизации игрового пространства воспользуемся алгоритмом растеризации. Поскольку результатом растеризации является карта высот, разбиение воксельного пространства на слои методом, описанным в оригинальном алгоритме, невозможно. Произведем модификацию алгоритма так, чтобы слои представляли собой области пространства, недостижимые из других областей. В этом случае результирующая навигационная сетка, получаемая на 5-м шаге алгоритма, будет представлять собой объединение сеток, полученных на 4-м шаге для каждого отдельно взятого слоя. Для этого поменяем местами шаги 2 и 3, т. е. произведем сначала обработку растровой сетки, а затем выполним извлечение из обработанной сетки слоев.

Результатом обработки слоя является двумерный массив, содержащий логические переменные, где истинному значению соответствует область игрового пространства, в которой навигационный агент может находиться, а ложному — непроходимая область. Произведем разбиение обработанной растровой сетки на слои следующим образом:

1. Создадим вспомогательный массив L с размером, совпадающим с размерами растровой сетки. В этом массиве будем хранить для каждой ячейки индекс слоя, к которому эта ячейка принадлежит. Установим каждой ячейке массива значение -1. Установим счетчик i равным 0. Также создадим массив V, содержащий информацию о том, была ли посещена ячейка или нет.

2. Для каждой непосещенной ячейки определим слой, к которому она принадлежит. Если ячейка является проходимой, то выполним заливку достижимых из текущей ячейки ячеек, помечая их как посещенные и устанавливая им в массиве L значение i. При завершении каждого применения алгоритма заливки будет увеличивать i на 1.

3. После того, как все ячейки посещены, определим границы слоев, найдя для каждого слоя ячейки с минимальным и максимальным значениями ширины и высоты.

4. Создадим слои определенных на 3-м шаге размеров, добавляя в слой k ячейки с индексом в массиве L, равным k, как проходимые, а остальные ячейки — как препятствия.

Определим в каждом слое контуры многоугольников, затем упростим контуры при помощи алгоритма Рамера-Дугласа-Пекера, после чего выполним построение графа ячеек и порталов из полученных контуров, используя алгоритм [8]. Объединение графов ячеек и порталов всех слоев и будет результирующей навигационной сеткой.

4.4. Алгоритм генерации навигационной сетки из нерегулярной четырехугольной сетки в шестиугольных ячейках

В работе [3] общая нерегулярная четырехугольная сетка формируется из шестиугольных ячеек, где каждая ячейка содержит свою нерегулярную четырехугольную сетку. Каждая

шестиугольная ячейка хранит информацию о соседних шестиугольниках, а доступ к граничным четырехугольникам соседних шестиугольников осуществляется за $O(1)$.

Построение навигационной сетки из сетки [3] осуществим следующим образом:

1. Растеризация геометрии игрового окружения, участвующей в формировании навигационной сетки.

2. Формирование проходимой плоскости при помощи проецирования нерегулярной четырехугольной сетки на растровую сетку, полученную на шаге 1.

3. Связывание соседних четырехугольников для получения навигационной сетки.

На шаге 2 каждой вершине нерегулярной четырехугольной сетки поставим в соответствие ячейку на растровой сетке, содержащей информацию о высоте, т. е. произведем растеризацию нерегулярной четырехугольной сетки. Затем рассмотрим пары соединенных вершин и установим непроходимыми те пары, значения высоты которых образуют угол наклона с горизонтальной плоскостью больший, чем некоторое заданное значение ε. Получим для каждого четырехугольника информацию о проходимости через каждое из четырех ребер.

Поскольку четырехугольники в шестиугольных ячейках не содержат информацию о соседних четырехугольниках (кроме тех, что находятся на границах ячеек), соединим в каждом шестиугольнике четырехугольники. Выполним поиск для каждого четырехугольника тех четырехугольников, которые содержат пару индексов вершин, присутствующих в исходном четырехугольнике.

Заключение

В результате выполнения работы среди множества алгоритмов генерации навигационных сеток были выбраны алгоритмы Recast, NEOGEN и равномерная прямоугольная сетка. Была предложена модификация алгоритма NEOGEN для построения алгоритмом сеток в недостижимых из исходной точки, выбираемой пользователем, областях игрового пространства. Был также предложен алгоритм генерации навигационных сеток из нерегулярных четырехугольных сеток в шестиугольных ячейках.

Литература

1. Какие типы навигации существуют? – Режим доступа: https://okami.group/all-about-navigation (дата обращения: 04.04.2023).

2. Навигационная сетка. – Режим доступа: https://wiki5.ru/wiki/Navigation_mesh (дата обращения: 04.04.2023).

3. Чернышов Н. М. Реализация алгоритма процедурной генерации нерегулярной четырехугольной сетки на плоскости на основе регулярной треугольной сетки в правильных шестиугольниках / Н. М. Чернышов, О. В. Авсеева // Системы управления и информационные технологии. – В.: Научная книга, 2021. – С. 88-95.

4. Kallmann M. Dynamic and robust Local Clearance Triangulations // ACM Transctions on Graphics 33.

5. Kallmann M. Fully dynamic constrained Delaunay triangulations / M. Kallmann, H. Bieri, D. Thalmann // In Geometric Modeling for Scientific Visualization. – Germany, 2003. – P. 241-257.

6. Van Toll W. G. Navigation meshes for realistic multi-layered environments / W. G. Van Toll, A. F. Cook IV, R. Geraerts // In proceedings of IEEE/RSJ international conference on Intelligent Robots and Systems. – P. 3526-3532.

7. Mononen M. Recast navigation: navigation-mesh toolset for games. Режим доступа: http://recastnav.com/ (дата обращения: 09.04.2023).

8. Oliva R. Automatic Generation of Suboptimal NavMeshes / R. Oliva, N. Pelechano // In proceedings of 4th International Conference on Motion in Games. – P. 328-329.

9. Oliva R. NEOGEN: Near optimal generator of navigation meshes for 3D multi-layered environments / R. Oliva, N. Pelechano // Computers & Graphics 37. – P. 403-412.

10. Pettre J. A navigation graph for real-time crowd animation on multilayered and uneven terrain / J. Pettre, J. Laumond, D. Thalmann // In proceedings of 1st International Workshop on Crowd Simulation. – P. 81-89.

11. Unreal Development Kit Navigation Mesh Reference. Режим доступа: https://docs.unrealengine.com/udk/Three/NavigationMeshReference.html (дата обращения: 04.04.2023).

12. Van Toll W. A comparative study of navigation meshes / W. Van Toll, R. Triesscheijn, M. Kallmann, R. Oliva, N. Pelechano, J. Pettre, R. Geraerts // MIG ’16 – 9th International Conference on Motion in Games. – San Francisco, 2016. – P. 91-100.

13. Oliva R. ANavMG: automatic generation of suboptimal NavMeshes for 3D virtual environments / R. Oliva, N. Pelechano. – Barcelona, 2012.

14. Rasterization: a practical implementation. Режим доступа: https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation/overview-rasterization-algorithm.html (дата обращения: 07.04.2023).

15. Mononen M. Automatic Navmesh Generation via Watershed Partitioning // AIGameDev master class slides. – Режим доступа: https://9631ec60-a-62cb3a1a-s-sites.googlegroups.com/site/recastnavigation/MikkoMononen_RecastSlides.pdf (дата обращения: 06.04.2023).

16. Meijster A. A general algorithm for computing distance in linear time / A. Meijster, J.B.T.M. Roerdink, W.H. Hesselink // Mathematical Morphology and its Applications to Image and Signal Processing. – Kluwer Acad. Publ., 2000. – P. 331-340.

17. Vincent L. Watersheds in digital spaces: an efficient algorithm based on immersion simulations / L. Vincent, P. Soille // IEEE Transactions on Pattern Analysis and Machine Intelligence. – P. 583-598.

18. Simplify polylines with the Douglas-Peucker algorithm. Режим доступа: https://towardsdatascience.com/simplify-polylines-with-the-douglas-peucker-algorithm-ac8ed487a4a1 (дата обращения: 16.04.2023).

19. Eberly D. Triangulation by Ear Clipping // Geometric tools. – Redmond, 2002. – Режим доступа: https://www.geometrictools.com/Documentation/TriangulationByEarClipping.pdf (дата обращения: 18.04.2023).

УДК 004

ГЕНЕРАЦИЯ ИГРОВОГО ПРОСТРАНСТВА С ПОМОЩЬЮ МЕТОДА BSP

И. Д. Чикунов
Воронежский государственный университет

Введение

Мировая игровая индустрия стремительно развивается, привнося не только новые игры, но и целые жанры.

Один из таких жанров - Roguelike, ключевой особенностью которого является генерируемые случайным образом уровни и игровое окружение. В случае поражения игрок начинает с начала. Для генерации уровней, окружения, элементов используются различные алгоритмы, которые решают те или иные игровые задачи.

Игры данного жанра интересны тем, что каждый раз игрок сталкивается с совершенно новым уровнем, который, возможно, никогда не повторится, что делает опыт уникальным. Также возможной частью является соревновательный аспект. Анализируя данный жанр, можно сказать, что он может быть объединен со многими другими, оставляя простор для воображения.

Методы, алгоритмы, разработки, используемые при создании игр, могут быть также использованы в других сферах, а постоянное развитие данной сферы стимулирует продвижение компьютерной графики.

1. Обзор алгоритмов для генерации

Для создания процедурной генерации уровней используется большой спектр алгоритмов, поэтому мы рассмотрим некоторые из них. Представленные ниже алгоритмы, за исключением метода BSP, имеют относительно узкую направленность, связанную с генерацией пространства, в то время как метод BSP используется в более широком

1.1. Алгоритм Эйлера

Алгоритм Эйлера [1] позволяет создавать лабиринты, имеющие только один путь между двумя точками. Сам по себе алгоритм очень быстр и использует память эффективнее, чем другие популярные алгоритмы (такие как Prim и Kruskal), требуя памяти пропорционально числу строк. Это позволяет создавать лабиринты большого размера при ограниченных размерах памяти. Сам алгоритм довольно громоздкий, если описывать его подробно, но тем не менее его суть заключается в объединении ячеек, не разделенных стеной в одно множество ячеек, причем заполнение происходит построчно сверху-вниз.

1.2. Алгоритм туннелирования

Этот алгоритм позволяет создавать уровни методом “прокапывания”, т.е. каждая последующая комната соединена с предыдущей с помощью коридоров. Для того чтобы повлиять на вариативность генерации используется стоимость прохода по разным тайлам. Также для того, чтобы комнаты были не обязательно прямоугольными, можно задать их генерацию из нескольких прямоугольников, соединенных вплотную [2].

1.3. Алгоритм BSP

Двоичное разбиение пространства [3] (англ. binary space partitioning) — метод рекурсивного разбиения евклидова пространства в выпуклые множества и гиперплоскости. В результате объекты получают представление в виде структуры данных, называемой BSP-деревом.

Данный метод применяется в алгоритмах отрисовки сцены, сортировке объектов относительно наблюдателя, в том числе отсечение невидимых поверхностей для уменьшения

количества объектов для обработки, поиска столкновений, случайной генерации объектов. Для каждой цели алгоритм может сильно варьироваться в зависимости от поставленной задачи.

BSP - алгоритмы используются и по сей день в графике и игровой индустрии.

Более подробное описание реализации метода для генерации игрового пространства и принципа его работы будет изложено ниже.

2. Метод BSP

Основным элементом работы метода является построение бинарного сбалансированного дерева. Способ построения и последующее его использование полностью зависит от поставленной задачи, но главный принцип остается неизменным.

1.1. BSP дерево

В *BSP-дереве* каждый узел связан с разбивающей прямой или плоскостью в 2-мерном или 3-мерном пространстве соответственно. При этом все объекты, лежащие с фронтальной стороны плоскости, относятся к фронтальному узлу, а все объекты, лежащие с оборотной стороны плоскости, относятся к оборотному узлу. Для определения принадлежности объекта к фронтальной или оборотной стороне разбивающей прямой или плоскости необходимо исследовать положение каждой его точки. Положение точки относительно плоскости определяется скалярным произведением нормали плоскости и координат точки в однородных координатах. Возможно три случая:

1. Скалярное произведение больше 0 — точка лежит с фронтальной стороны плоскости.
2. Скалярное произведение равно 0 — точка лежит на плоскости.
3. Скалярное произведение меньше 0 — точка лежит с обратной стороны плоскости.

Если для всех точек объекта скалярное произведение больше или равно 0, то он относится к фронтальному узлу. Если для всех точек объекта скалярное произведение меньше или равно 0, то он относится к оборотному узлу. Если скалярные произведения для точек объекта имеют разный знак, то он рассекается разбивающей плоскостью так, чтобы полученные объекты лежали только с фронтальной или только с оборотной стороны. Для каждого подузла *BSP-дерева* справедливо вышеприведенное утверждение, с тем исключением, что рассмотрению подлежат только те объекты, которые принадлежат к фронтальной или оборотной стороне разбивающей плоскости родительского узла.

В зависимости от поставленной задачи оптимальность метода варьируется от двух взаимоисключающих факторов:

1. Получение как можно более сбалансированного дерева (когда для любого узла количество граней в правом поддереве минимально отличается от количества граней в левом поддереве); это обеспечивает минимальную высоту дерева (и соответственно наименьшее количество проверок).
2. Минимизация количества разбиений; одним из негативных свойств BSP-деревьев является разбиение граней, приводящее к значительному увеличению их общего числа и, как следствие, к росту затрат (памяти и времени) на изображение сцены. Так как эти критерии являются взаимоисключающими, обычно выбирается некоторый компромиссный вариант; например, в качестве критерия выбирается сумма высоты дерева и количества разбиений с заданными весами.

1.2. Генерация пространства с помощью BSP дерева

Пусть нам дана область, которую мы должны разбить на комнаты.

В качестве оптимизационных составляющих выставим ограничение в количестве итераций и выставим ограничения на минимальные размеры комнат, конечно, в финальном варианте эти

параметры можно будет менять для того, чтобы продемонстрировать работоспособность метода.

1. Проведем первую прямую, разделив тем самым нашу область на 2 подобласти.

На рисунке 1 представлен результат работы первой итерации

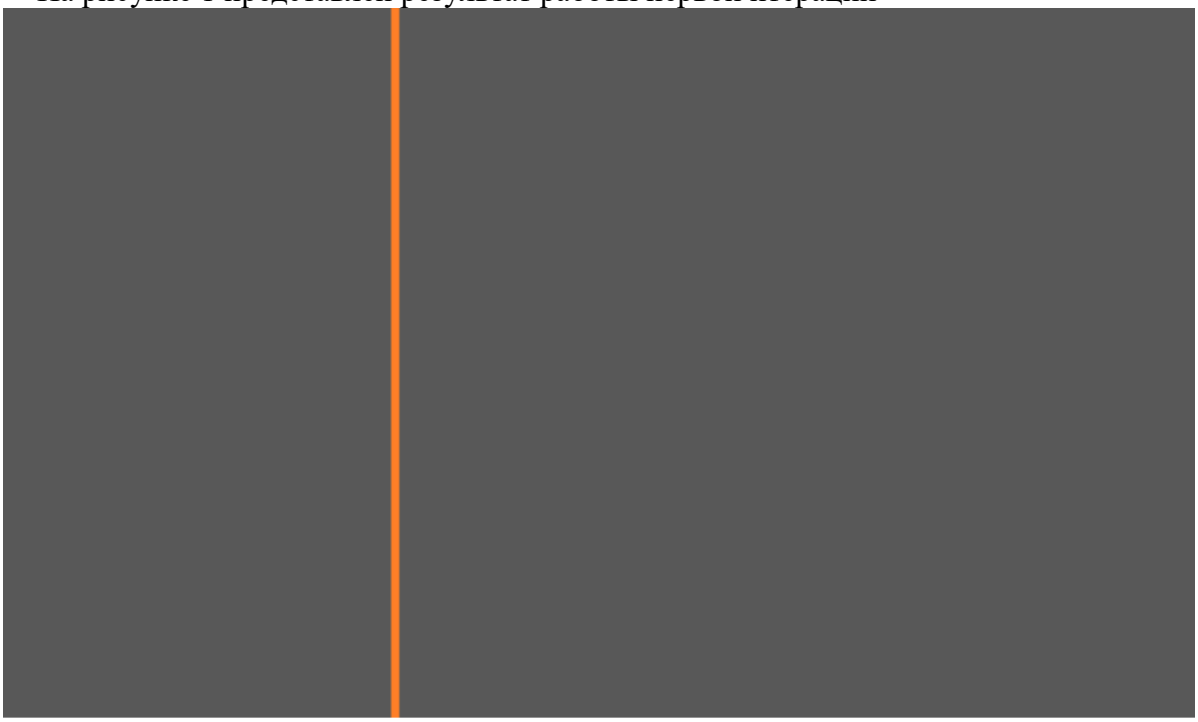


Рис. 1 Первая возможная итерация

Первую и последующие прямые мы ставим случайным образом, но вертикально или горизонтально, причем так, чтобы у разделенных областей хватало места для размещения комнаты с учетом минимального задаваемого ее размера.

2. Пример возможной второй итерации:

На рисунке 2 представлен результат второй итерации.

Рис. 2 Вторая возможная итерация

3. Метод оптимизации в нашей задаче заключается в том, что области которые получаем, мы добавляем в очередь на разбиение. Таким образом у нас создается сбалансированное дерево генерации, т.е. :

На рисунке 3 представлен результат работы несбалансированного метода.

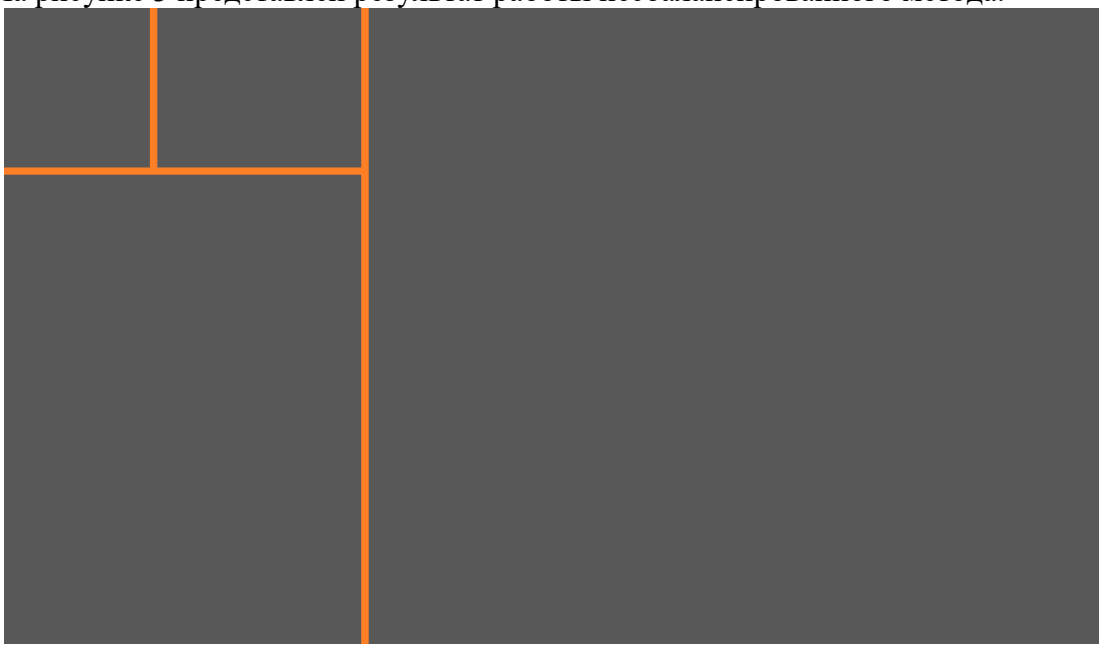


Рис. 3 Пример несбалансированного дерева

При методе генерации по левому направлению узлов может сложиться ситуация, при которой имея малые ограничения на минимальный размер комнаты и небольшое количество итераций наша область будет сгенерирована очень “неслучайно”, что будет проглядываться в том, что некоторые области сильно раздроблены, в то время как остальные будут большими.

На рисунке 4 представлен результат работы сбалансированного метода.

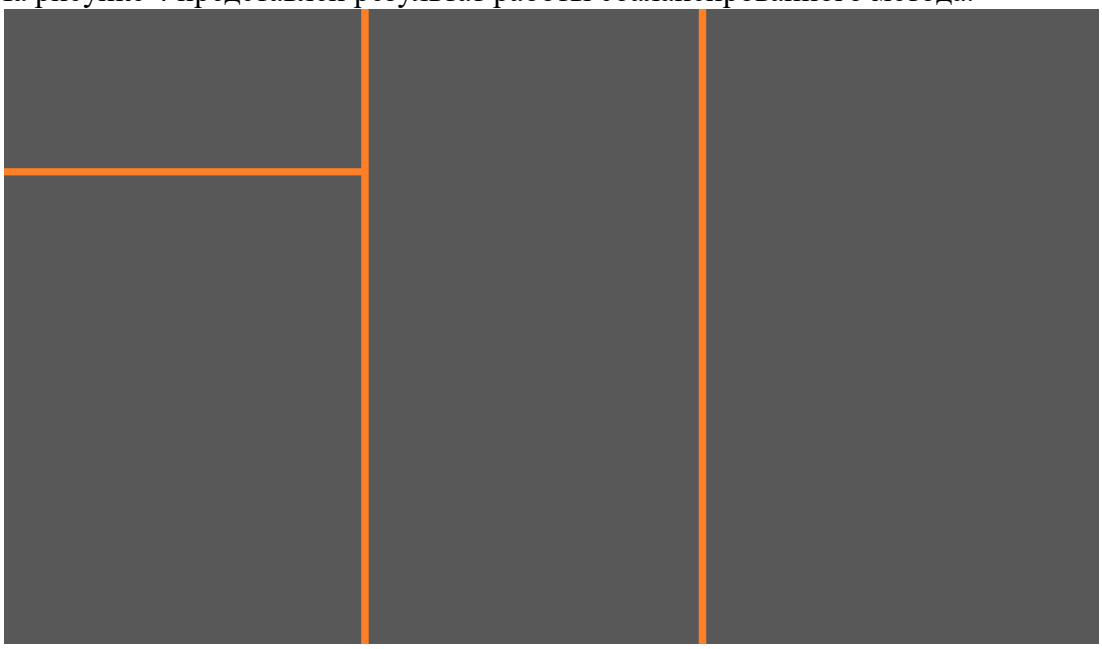


Рис. 4 Возможная итерация

4. Финальное разбиение может иметь вид:

На рисунке 5 представлен конечный результат работы алгоритма.

Рис. 5 Финал теоретического примера

Таким образом мы получаем конечный вид BSP-дерева. Теперь если мы рассмотрим все листья (элементы дерева, не имеющие потомков), то они будут являться возможными комнатами. Способ заполнения этих пространств уже может разниться в зависимости от цели, в нашем случае мы будем использовать пространство максимально, но с небольшими отступами.

В случае разбиения N-мерного пространства размерность разделителя будет N-1, таким образом при разбиении плоскости разделителем является прямая, а вернее отрезок, для трехмерного пространства - плоскость.

Заключение

Метод BSP [4] является универсальным инструментом работы с цифровой графикой, использующийся как для обнаружения столкновений графических объектов, так и сортировке визуальных объектов в порядке удаления от наблюдателя. Генерация игрового пространства – лишь один из вариантов интерпретации алгоритма.

Литература

1. Алгоритм Эллера для генерации лабиринтов / Хабр. -URL: https://habr.com/ru/post/176671/ (дата обращения: 14.03.2023)
2. Процедурная генерация в roguelike. -URL: https://www.pvsm.ru/razrabotka-igr/280096 (дата обращения: 14.03.2023)
3. Двоичное разбиение пространства - Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf (2000), Computational Geometry (2nd revised ed.), Springer-Verlag, ISBN 3-540-65620-0 Chapter 12: Binary Space Partition: pp.259–267.
4. BSP-дерево - Викиконспекты. -URL: https://neerc.ifmo.ru/wiki/index.php?title=BSP-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE (дата обращения: 14.03.2023)

УДК 004.42

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ МЕДИЦИНСКОГО УЧРЕЖДЕНИЯ

Е. М. Шелудякова

Воронежский государственный университет

Введение

Медицинские организации являются операторами персональных данных своих пациентов. Они принимают непосредственное участие в сборе, систематизации, накоплении и хранении такой информации. Без специализированной системы работники медицинских учреждений вынуждены вручную заполнять документы, а также возникает риск отсутствия систематизации в их хранении.

В связи с этим для облегчения управления и контроля за данными необходима специальная информационная система, позволяющая упростить операции с ними.

1. Постановка задачи

Работа медицинского регистратора связана, как с записью клиентов к врачу, так и с взаимодействием с медицинским работником: передача данных пациента, сообщение пациенту о результатах. В свою очередь у врача существует как работа непосредственно с клиентом, так и работа с заполнением документов, данных об осмотре и его результате.

Для организации данных задач требуется разработать информационную систему, обеспечивающую:

1. Автоматизированную запись к врачу.
2. Добавление, модификацию, удаление и хранение данных о пациентах, врачах, анамнезе, диагнозе и ходе осмотра.
3. Автоматизированную передачу данных о пациенте врачу.
4. Заполнение шаблонов стандартизированных отчетов о ходе осмотра врачом.

В системе необходимо предусмотреть три категории пользователей: администратор, медицинский регистратор, лечащий врач.

Администратор должен иметь полный доступ ко всем функциям системы. Медицинский регистратор должен иметь возможность вводить информацию о пациентах, передавать информацию лечащему врачу, отправлять пациентам результаты исследований и осмотров. Лечащий врач должен заполнять стандартные документы, содержащие информацию об осмотре

2. Средства реализации

Для хранения информации о пациентах и деталях заболеваний необходимо разработать базу данных.

В качестве РСУБД был использован Postgresql – объектно-реляционная система управления базами данных, поддерживающая многие из возможностей стандарта SQL:2011.

Для разработки серверной и клиентской частей была выбрана Java в сопровождении

фреймворка GWT, его основная особенность – компилятор из Java в JavaScript, что позволяет реализовать почти всю разработку клиентской части на основе Java и только на последнем этапе создать соответствующие JavaScript, HTML и CSS.

3. Описание разработки информационной системы

3.1. Описание структуры данных

Для хранения информации о работниках медицинской организации, пациентах, визитах, документах и услугах была разработана база данных (рис. 1).

Структура БД состоит из 15 таблиц.

PERSON – сущность «Человек». Хранит основную информацию о человеке – ФИО, телефон, email, пол.

GENDER_ITEM – сущность «Пол». Хранит информацию о полах людей.

OFFICIAL_DOCUMENTS – сущность «Официальные документы». Хранит информацию о полях документа, является ли документ актуальным, наименование документа.

ADDRESS – сущность «Адрес». Хранит информацию об адресе человека: страна, город, улица, подъезд, дом, квартира.

CHRONICAL_DISEASE – сущность «Хроническое заболевание». Хранит название хронического заболевания.

CHRONIC_DISEASE_LINK – сущность для связи между пациентом и хроническим заболевание, обеспечивающая связь «многие ко многим».

VISIT – сущность «Визит». Хранит информацию о пациенте, на которого заведен визит, времени создания визита, создателе визита, времени окончания визита, является ли визит открытым в данный момент.

PATIENT – сущность «Пациент». Хранит информацию о человеке, который является пациентом, росте, весе, группе крове, резус-факторе.

STAFF – сущность «Работник». Хранит информацию о должности работника, рабочем стаже, работает ли он в данный момент.

SERVICE – сущность «Услуга». Хранит информацию о наименовании услуги, времени записи на услугу, визите, во время которого она должна быть выполнена, враче, который должен ее оказать.

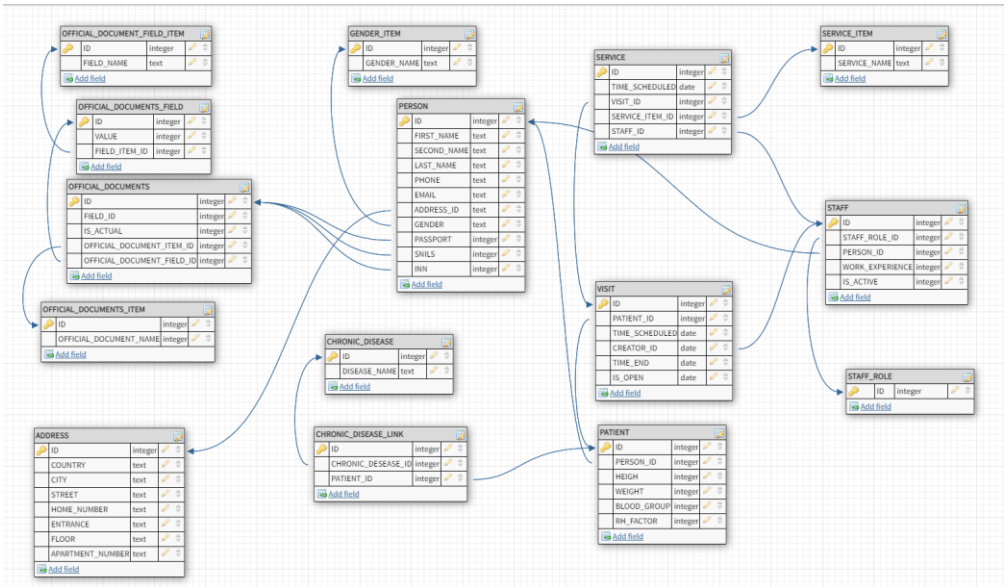


Рис. 1 Схема БД

3.2. Сервер

Серверная часть приложения написана с использование языка Java. Приложение построено по многослойной архитектуре (рис. 2).

Данная архитектура позволяет сбалансировать нагрузку на каждый из уровней, распределяя ресурсы наиболее эффективно.

Логика приложения хранится и выполняется централизованно, что позволяет обеспечить целостность данных и логики, а также упрощает сопровождение и поддержку.

Архитектура позволяет эффективно организовать процесс обработки ИС, так как для реализации каждого уровня необходимы собственные инструментальные средства и группы специалистов, независимые друг от друга.

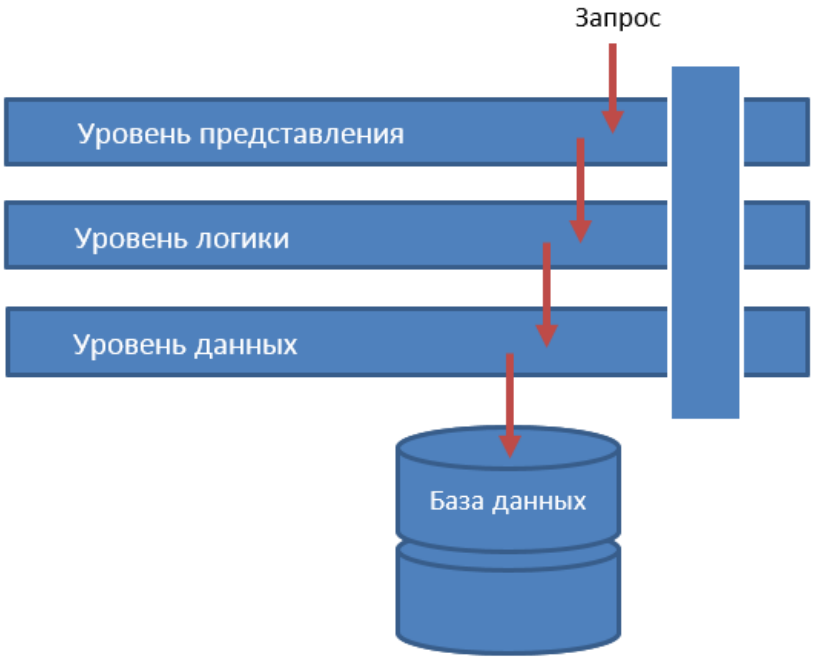


Рис. 2. Многослойная архитектура

Уровень баз данных – содержит все базы данных. Он отвечает за выполнение операций CRUD с помощью SQL запросов.

Уровень хранения данных – совокупность всех моделей и DTO-объектов составляет уровень хранения данных.

Уровень бизнес-логики. Классы, реализующие операции на уровне бизнес-логики называются сервисными классами, и реализуют базовые операции с данными: добавление, получение, обновление, удаление. Этот слой использует службы, предоставляемые уровнями доступа к данным, а также отвечает за валидацию и авторизацию. Этот слой знает все о нижестоящем, и следственно может использовать его для корректной работы;

Уровень представления данных. Он получает данные от клиента, затем передает их на дальнейшую обработку уровню бизнес-логики. Также этот уровень отвечает за отправку данных в понятном клиенту формате.

3.3. Клиент

Проект создан с учетом MVP паттерна (рис. 2).

Model (Модель) работает с данными, проводит вычисления и руководит всеми бизнес-процессами.

View (Вид или представление) показывает пользователю интерфейс и данные из модели.

Presenter (Представитель) служит прослойкой между моделью и видом.

Элемент Presenter при этом выполняет роль посредника между View и Model, а также позволяет эффективно управлять событиями клиентской стороны. Этот подход позволяет создавать абстрактные представления.

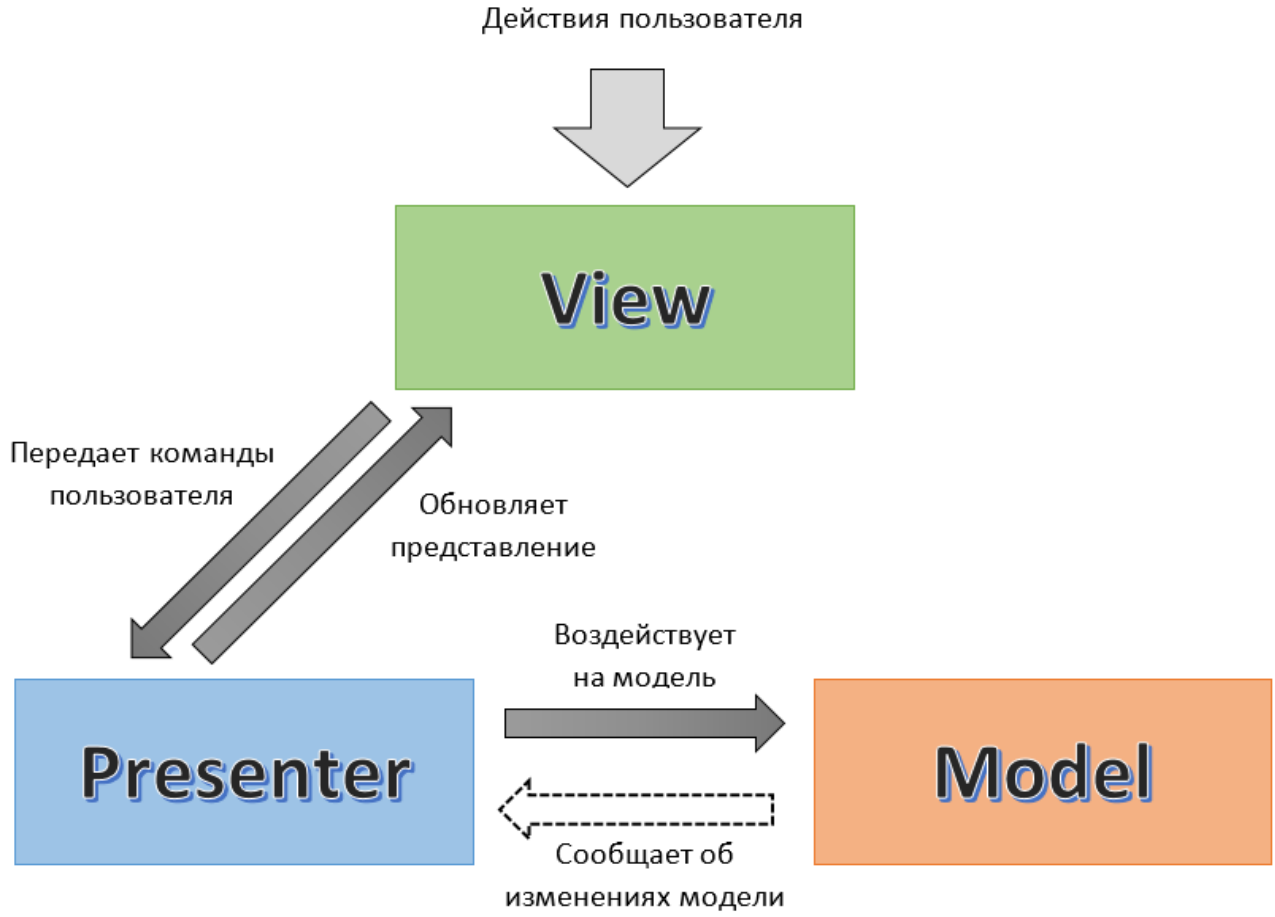


Рис. 3. MVP паттерн

Клиентская часть также написана с использование языка Java, код которой в последствии компилируется в JavaScript при помощи GWT фреймворка для отображения у пользователя. Логика работы фреймворка включает в себя удаление из кода неиспользуемых классов, методов, полей и сокращение имен Javascript.

Из-за этого преимущества больше не нужно включать библиотеки Ajax в проект Javascript. Также возможно задавать подсказки во время компиляции кода.

Основу GWT можно разделить на три основные части:

Компилятор GWT Java to JavaScript – это самая важная часть GWT. Компилятор GWT используется для перевода всего кода приложения, написанного на Java, в JavaScript.

Библиотека эмуляции JRE – GWT включает в себя библиотеку, которая эмулирует подмножество библиотек времени выполнения Java.

Библиотека создания пользовательского интерфейса GWT. Эта часть GWT состоит из множества частей, которые включают в себя фактические компоненты пользовательского интерфейса, поддержку RPC, управление историей и многое другое. Эта библиотека является одним из основных инструментов, для разработки интерфейса пользователя.

Также приложение, написанное на GWT, является кросс-браузерным. GWT автоматически генерирует код JavaScript, подходящий для каждого браузера.

Для связи между отображаемыми элементами интерфейса был использован Event Bus.

Шина событий – это шаблон проектирования, который обеспечивает связь между слабо связанными компонентами по принципу «публикатор события-подписчик на событие».

Компонент может отправить сообщение на шину событий, не зная конечного получателя. С другой стороны, компонент может прослушать сообщение на шине событий и решить, что с ним делать, не имея никакой информации о его отправителе.

Заключение

В ходе работы была разработана информационная система, обеспечивающая хранение и систематизацию данных в медицинском учреждении. Её использование помогает быстро, качественно и точно заполнять и обрабатывать данные о пациенте, оптимизировать и сократить работу с бумажными носителями.

Система предусматривает несколько категорий пользователей: администратор, медицинский регистратор, лечащий врач.

В качестве архитектуры использовалась клиент-серверная архитектура.

При разработке использовалось только свободное ПО, отвечающее требованиям разработки, что обеспечивает дальнейшее свободное сопровождение и модификацию элементов информационной системы.

Литература

1. Эккель, Б. Философия Java. – 4-е полное изд. – СПб.: Питер, 2018. – 1168 с.
2. Хорстман, Кей С. Java. Библиотека профессионала. Основы. – 11-и изд.: Пер. с англ. – СПб.: ООО «Диалектика», 2019. – 864 с.
3. Моргунов Е. П. PostgreSQL. Основы языка SQL: учеб. Пособие / Е. П. Моргунов. – 1-е изд. – СПб.: БХВ-Петербург, 2018. – 337 с.
4. Макконнелл, С. Совершенный код. Мастер-класс – Пер. с англ. – М.: Издательство «Русская редакция», 2010. – 896 стр.
5. Фримен Э., Робсон Э., Сьерра К., Бейтс Б. Head First. Паттерны проектирования. Обновленное юбилейное издание. – СПб.: Питер, 2018. – 656 с.

УДК 519.81

ЛУЧШАЯ СТРАТЕГИЯ ДЛЯ ПОЛУЧЕНИЯ МАКСИМАЛЬНОЙ ПРИБЫЛИ КАК ЗАДАЧА ПРИНЯТИЯ РЕШЕНИЯ В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ

О.С. Шершнева, Е.М. Аристова

Воронежский государственный университет

Введение

Наиболее характерным типом задач принятия решений являются задачи в условиях неопределенности [1]. Решение таких задач представляет наибольшую трудность, так для них невозможно сделать достоверный прогноз или оценить вероятность возникновения различных объективных условий. Решение таких задач нельзя также свести к составлению и решению математических моделей, как это делается для задач, решаемых в условиях определенности.

В реальных случаях, когда вначале кажется, что отсутствуют какие-либо оценки вероятностей достижения различных результатов, специалист по принятию решений обычно прилагает максимальные усилия для получения информации об этих вероятностях путем проведения специального исследования, и, как правило, это ему удается. Однако возможны и случаи, когда оценки вероятностей совершенно неизвестны [2].

Для задач принятия решений в условиях неопределенности характерна большая неполнота и недостоверность информации, многообразие и сложность влияния социальных, экономических, политических, технических и других факторов. Эти обстоятельства не позволяют построить адекватные математические модели решения задач по определению оптимального решения. Поэтому основную роль в поиске оптимального или приемлемого решения выполняет человек. Большое значение в данном случае имеют индивидуальные особенности руководителя. Формальные методы используются человеком в процессе формирования решений в качестве вспомогательных инструментов. Изложенное показывает, что задача принятия решений в условиях неопределенности является более общей и включает в себя частный случай принятия решений в условиях определенности и вероятностной определенности [3].

Принятие решения в условиях неопределенности основано на том, что вероятности различных вариантов ситуаций развития событий субъекту, принимающему решение, неизвестны.

В этом случае при выборе альтернативы принимаемого решения субъект руководствуется, с одной стороны, своим рисковым предпочтением, а с другой, – соответствующим критерием выбора из всех альтернатив по составленной им «матрице решений» (под *матрицей принятия решений* обычно понимается список значений в строках и столбцах, который позволяет аналитику систематически идентифицировать, анализировать и оценивать эффективность взаимосвязей между наборами значений и информацией [4]).

В качестве основных критериев, используемых в процессе принятия решений в условиях неопределенности, выделяют:

- критерий Вальда (критерий «максмина»);
- критерий «максмакса»;
- критерий «минмакса»;
- критерий Байеса-Лапласа;
- критерий Гурвица (критерий «оптимизма-пессимизма» или «альфа-критерий»);

- критерий Сэвиджа (критерий потерь от «минимакса»);
- критерий Ходжа-Лемана;
- критерий Гермейера;
- критерий произведений.

В данной статье рассматриваются 2 критерия: критерий Вальда и критерий Гурвица. Основное отличие критерия Гурвица от критерия Вальда заключается в том, что критерий Гурвица учитывает как пессимистический, так и оптимистический подход к ситуации.

Критерий Гурвица используется, если требуется остановиться между линией поведения в расчете на худшее и линией поведения в расчете на лучшее, поэтому его часто называют критерием «пессимизма-оптимизма».

1. Постановка задачи

Постановка задач в условиях неопределённости и риска заключается в следующем: человек выбирает какие-либо действия в мире, где на полученный результат (исход) действия влияют случайные события, неподвластные человеку, но, имея некоторые знания о вероятностях этих событий, человек может рассчитать наиболее выгодную совокупность и очерёдность своих действий, при этом иногда неизвестны даже вероятности, тогда решение лица, принимающего решение (ЛПР), зависит от его отношения к риску, который можно заранее проанализировать. Принимая решение в условиях неопределённости, ЛПР рискует.

Риск – это сложное явление, характеристиками которого являются: неизвестность (неопределённость) будущих результатов, вероятность отрицательных результатов деятельности, их величина, а также значимость результатов для принимающего решение.

2. Методы принятия решения

Рассмотрим вначале *критерий Вальда* [5]. Данный критерий предполагает, что из всех возможных вариантов «матрицы решений» выбирается та альтернатива, которая из всех самых неблагоприятных ситуаций развития события (минимизирующих значение эффективности) имеет наибольшее из минимальных значений (т.е. значение эффективности, лучшее из всех худших или максимальное из всех минимальных):

$$F_{maxmin} = max_i min_j u_{ij}, \quad (1)$$

где F_{maxmin} – средневзвешенная эффективность по критерию Вальда для конкретной альтернативы;

max_i – максимальное значение по min_j;

min_j – минимальное значение эффективности по конкретной альтернативе.

Критерием Вальда (критерием «максимина») руководствуются при выборе рисковых решений в условиях неопределенности, как правило, субъект, не склонный к риску или рассматривающий возможные ситуации, как пессимист.

Рассмотрим теперь *критерий Гурвица* [6], который позволяет руководствоваться при выборе решения в условиях неопределенности некоторым средним результатом эффективности, находящимся в поле между значениями по критериям «максимакса» и «максимина» (поле между этими значениями связано посредством выпуклой линейной функции). Оптимальная альтернатива решения по критерию Гурвица определяется на основе формулы:

$$F_H = \max_i \left\{ \alpha \min_j u_{ij} + (1-\alpha) \max_j u_{ij} \right\}, \quad (2)$$

где F_H – средневзвешенная эффективность по критерию Гурвица для конкретной альтернативы;

α – альфа-коэффициент, принимаемый с учетом рискового предпочтения в промежутке от 0 до 1 (значения, приближающиеся к нулю, характерны для субъекта, не склонного к риску; значение равное 0,5 характерно для субъекта, нейтрального к риску; значения, приближающиеся к единице, характерны для субъекта, склонного к риску);

max_j – максимальное значение эффективности по конкретной альтернативе;

min_j – минимальное значение эффективности по конкретной альтернативе.

Критерий Гурвица используют при выборе рисковых решений в условиях неопределенности те субъекты, которые хотят максимально точно идентифицировать степень своих конкретных рисковых предпочтений путем задания значения альфа-коэффициента.

3. Вычислительный эксперимент

Компания «Газировка» выпускает газированный напиток, который упаковывается в 40-литровые бочки. Напиток готовится в течение недели, и каждый понедельник очередная партия готова к употреблению. Однако, в одно из воскресений всю готовую к продаже партию пришлось выбросить. Секретный компонент, используемый для приготовления напитка, покупается в небольшой лаборатории, которая может производить каждую неделю в течение полугода (так налажено производство) только определенное количество этого компонента. Причем он должен быть использован в кратчайший срок.

Переменные затраты на производство одного литра напитка составляют 70 условных единиц (у.е.), продается он за 150 у.е. Однако, компания предвидит, что срыв поставок приведет к потере части покупателей в долгосрочной перспективе, а, следовательно, придется снизить цену на 30 у.е. За последние 50 недель каких-либо явных тенденций в спросе выявлено не было. Данные приведены в табл.1:

Таблица 1

Спрос на бочки в неделю	3	4	5	6	7
Число недель	5	10	15	10	10

Необходимо выбрать лучшую стратегию закупки бочек для получения максимальной прибыли.

Решение.

В начале недели можно изготавливать для последующей продажи от трех до семи бочек напитка. В целом каждое решение и его исходы примерно равны, но, принимая решение, невозможно контролировать исходы. Покупатели определяют их сами, поэтому исходы представляют собой также «фактор неопределенности».

Построим игровую матрицу. Стратегии компании – выпуск бочек в соответствии со спросом (3, 4, 5, 6 или 7). Стратегии «природы» – это спрос, который заранее неизвестен. Прибыль фирмы находится по формулам:

– если спрос больше или равен запасу бочек, то прибыль получается только от запаса:

$$F = (C - P) \cdot Z,$$

где *F* – прибыль фирмы;
C – цена;
P – переменные затраты на производство;
Z – запас;

– если спрос меньше запаса бочек, то спрос удовлетворяется:

$$F = (C - P) \cdot D,$$

где *F* – прибыль фирмы;
C – цена;
P – переменные затраты на производство;
D – спрос;

– остаток продается позже по сниженной цене:

$$O = (C - P - S) \cdot (Z - D),$$

где *F* – прибыль фирмы;
C – цена;
P – переменные затраты на производство;
D – спрос;
Z – запас;
S – скидка (%).

В итоге получим матрицу (табл.2):

Таблица 2

Игровая матрица

Бочки\спрос	**3**	**4**	**5**	**6**	**7**
3	240	240	240	240	240
4	290	320	320	320	320
5	340	370	400	400	400
6	390	420	450	480	480
7	440	470	500	530	560

Посчитаем предполагаемые вероятности спроса по формуле $K_i = \frac{S}{T}$, (3)

где K_i – вероятность i-го спроса;
S – число недель, соответствующее спросу на бочки в неделю;
T – общее число недель.

Получим, что по формуле вероятности имеют вид:
3 бочки – 5/50 = 0,1; 4 бочки – 10/50 = 0,2; 5 бочек – 15/50 = 0,3;
6 бочек – 10/50 = 0,2; 7 бочек – 10/50 = 0,2.

Для решения задачи воспользуемся критерием Вальда [7]**.** За оптимальную принимается чистая стратегия, которая в наихудших условиях гарантирует максимальный выигрыш, т.е. $F_{maxmin} = max_j \min_j u_{ij}$.

Получим расчетную таблицу (табл.3):

Таблица 3

Расчетная таблица

Критерий Вальда	**3**	**4**	**5**	**6**	**7**	**min**	**max**	$F_{maxmin} = max_i min_j u_{ij}$,
3	240	240	240	240	240	240	240	240
4	290	320	320	320	320	290	320	290
5	340	370	400	400	400	340	400	340
6	390	420	450	480	480	390	480	390
7	440	470	500	530	560	440	560	440

Из таблицы можно сделать вывод, что для получения максимальной прибыли, равной 440 у.е., необходимо выбрать стратегию закупки семи бочек.

Применим теперь критерий Гурвица [8]. Критерий Гурвица является критерием пессимизма-оптимизма. В этом критерии используется та же игровая матрица (табл.2). За оптимальную принимается та стратегия, для которой выполняется соотношение (2). Например, для α = 0,5 расчетная таблица примет вид (табл.4):

Таблица 4

Расчетная таблица

Критерий Гурвица	**3**	**4**	**5**	**6**	**7**	**min**	**max**	$F_H = \max_i \left\{ \alpha \min_j u_{ij} + (1-\alpha) \max_i u_{ij} \right\}$
3	240	240	240	240	240	240	240	240
4	290	320	320	320	320	290	320	305
5	340	370	400	400	400	340	400	370
6	390	420	450	480	480	390	480	435
7	440	470	500	530	560	440	560	500

Из таблицы можно сделать вывод, что для получения максимальной прибыли, равной 500 у.е., необходимо выбрать стратегию закупки семи бочек.

Вывод

По критерию Вальда за оптимальную принимается стратегия, которая в наихудших условиях гарантирует максимальный выигрыш. Критерий Вальда ориентирует статистику на самые неблагоприятные условия.

Критерий Гурвица устанавливает баланс между случаями крайнего пессимизма и крайнего оптимизма путем взвешивания обоих способов поведения соответствующими весами (1 - α) и у, где 0 < α < 1.

Проведя вычислительный эксперимент, используя эти критерии, можно сделать вывод, что оба критерия являются самыми «осторожными» подходами к принятию решений и наиболее учитывающими все возможные риски.

Заключение

В работе были выполнены следующие задачи:

- рассмотрены основные понятия теории принятия решения;
- поставлена задача о нахождении лучшей стратегии для подсчета максимальной прибыли;
- решена задача в условиях неопределенности с использованием двух критериев – Вальда и Гурвица;
- проведен сравнительный анализ двух критериев.

Кандидат физико-математических наук, доцент кафедры вычислительной математики и прикладных информационных технологий факультета прикладной математики, информатики и механики Воронежского государственного университета, Аристова Екатерина Михайловна.

Литература

1. Блягоз, З.Ю. Информатика и вычислительная техника. Принятие решений в условиях риска и неопределенности / З.Ю. Блягоз, А.Ю. Попова // Вестник Адыгейского государственного университета. – 2006. – С. 53-55.
2. Бородачёв, С. М. Теория принятия решений: учебное пособие / С. М. Бородачёв. – Екатеринбург : Изд-во Урал, университета, 2014. – 124 с.
3. Зайцев, М.Г. Методы оптимизации управления для менеджеров / М.Г. Зайцев, С.Р. Филонович. – Москва: Дело, 2002. – 240 с.
4. Матрица принятия решений. – Режим доступа: https://fb.ru/article/467312/matritsa-prinyatiya-resheniy-vidyi-vozmojnyie-riski-provedenie-analiza-i-posledstviya. – (Дата обращения: 30.03.23).
5. Принятие решений в условиях риска и неопределённости. – Режим доступа: https://mbschool.ru/articles/prinyatie-resheniy-v-usloviyakh-riska-i-neopredelyennosti/. – (Дата обращения: 30.03.23).
6. Принятие решений в условиях риска и неопределенности. – Режим доступа: https://studme.org/335483/menedzhment/prinyatie_resheniy_usloviyah_risk a_neopredelennosti .– (Дата обращения: 25.03.23).
7. Решения в условиях определенности, риска и неопределенности. – Режим доступа: https://studyport.ru/referaty/ekonomika/1543-reshenija. – (Дата обращения: 31.03.23).
8. Силкина, Г.Ю. Теория принятия решений и управление рисками / Модели конфликтов, неопределенности, риска: учебное пособие / Г.Ю. Силкина. – Санкт-Петербург: Изд-во СПбГПУ, 2003. – 72 с.

УДК 004.932.2

ИССЛЕДОВАНИЕ АЛГОРИТМОВ СЕГМЕНТАЦИИ И РАЗРАБОТКА МОБИЛЬНОГО ANDROID-ПРИЛОЖЕНИЯ ДЛЯ ИХ ПРИМЕНЕНИЯ

И. Г. Шилова, Е. В. Трофименко

Воронежский государственный университет

Введение

Задача сегментации — одна из основных задач обработки и анализа изображений, позволяющая разделять изображения на области, в каждой из которых пиксели обладают определенным критерием схожести: например, в одной области будут лежать пиксели, обладающие примерно одинаковым значением яркости и/или цвета.

В работе [1] были исследованы такие алгоритмы как: метод К-средних, метод обнаружения контура, сегментация методом Оцу, а также предложен способ улучшения работы данных алгоритмов на изображениях с незначительными дефектами (царапинами на поверхности) – применение предобработки изображений путем повышения контрастности, увеличение резкости и применение сглаживания.

В данной статье проводится анализ двух дополнительных алгоритмов: метод водоразделов и метод выращивания регионов; анализируется время работы всех рассмотренных алгоритмов, а также рассматривается возможность внедрения рассмотренных алгоритмов в мобильное Android-приложение.

1. Алгоритмы сегментации

1.1. Метод водораздела

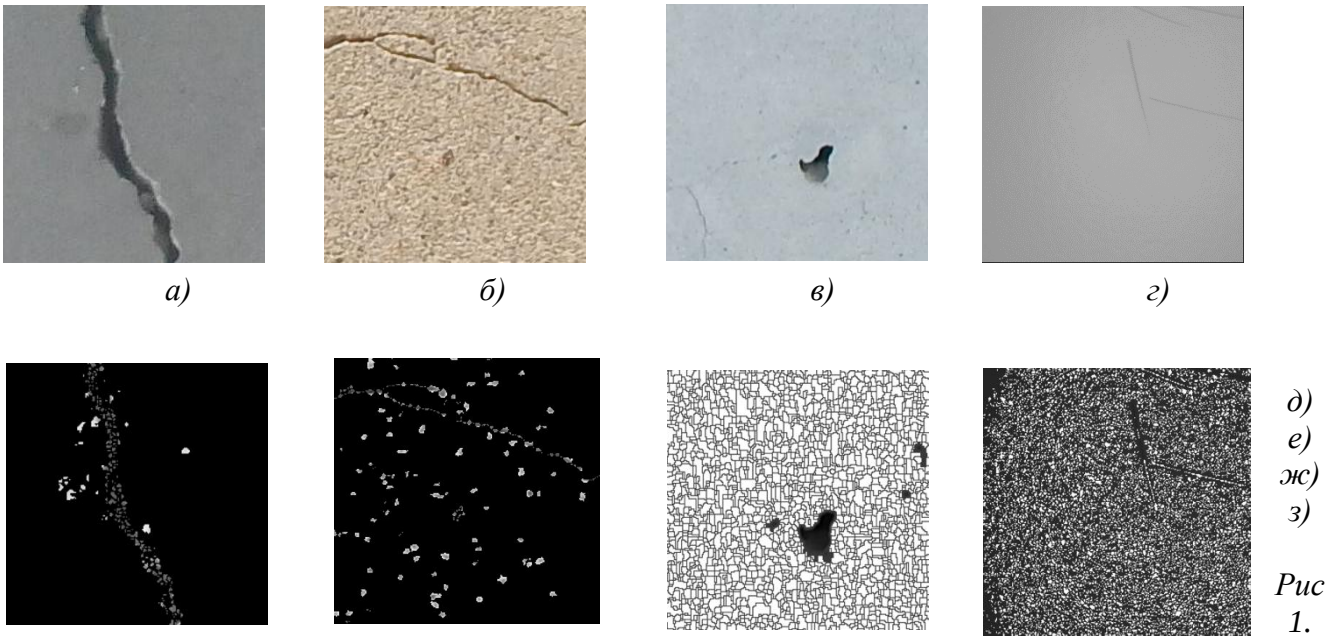
В методе водораздела [2][4] изображение рассматривается как некоторая карта местности, где значение яркости пикселя представляют собой значение высот. Благодаря таким расчетам, пиксели с высокой интенсивностью образуют «горы», а с низкой – «впадины». Полученная с такими расчетами карта местности начинает заполняться водой: впадины будут постепенно заполняться, сливаясь с соседними, при этом место слияния будет обозначаться границей водораздела – именно по этим границам осуществляется разбиение изображения на сегменты.

Рассмотрение работы алгоритма проводилось на изображениях, используемых в исследовании предыдущих алгоритмов. Результаты приведены на рис. 1.

Исходя из полученных результатов можно сделать вывод о том, что метод водоразделов частично справился с задачей выделения дефектов на тестовых изображениях. Можно заметить, что на рис. 1е) алгоритм выделил несколько микро-дефектов наряду с основным дефектом (трещиной), несмотря на выраженную текстуру поверхности. Также, на рис. 1з) метод водораздела, как и прочие рассмотренные ранее алгоритмы, не справился с выделением такого дефекта, как царапина.

Также стоит добавить, что из-за особенности реализации алгоритма, нанесение на

изображение границ мешает отфильтровать пиксели на картинке таким образом, чтобы на результирующем изображении остался только дефект.



а) *б)* *в)* *г)*

д)
е)
ж)
з)

Рис 1.

Результаты сегментации методом водораздела. а-г) Изображения до сегментации, д-з) результаты после сегментации.

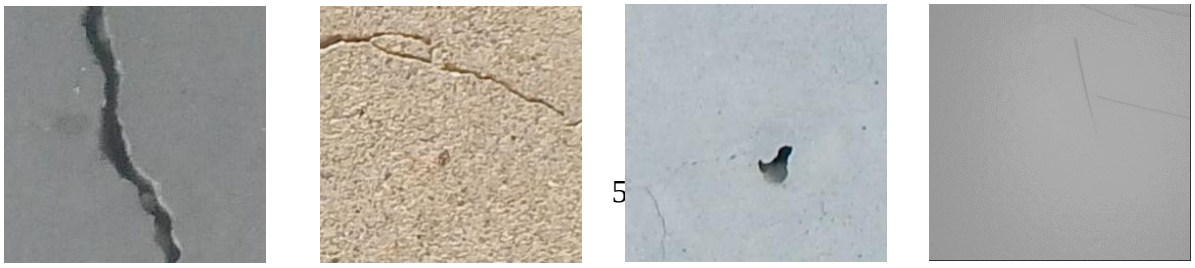
1.2. Метод выращивания регионов

Идея метода выращивания регионов [3] заключается в следующем: сначала необходимо выбрать центры будущих регионов (сегментов). Это можно сделать рандомным образом или по заранее определенным правилам. Далее итеративно к каждому региону присоединяются соседние незанятые пиксели изображения, которые удовлетворяют некоторому заранее определенному условию. В данном алгоритме для проверки такого условия будет вычисляться разница яркостей между пикселями, которая в свою очередь сравнивается с пороговым значением, которое задается пользователем заранее. Пороговое значение здесь – гиперпараметр, который требует подбора для получения лучшего результата сегментации.

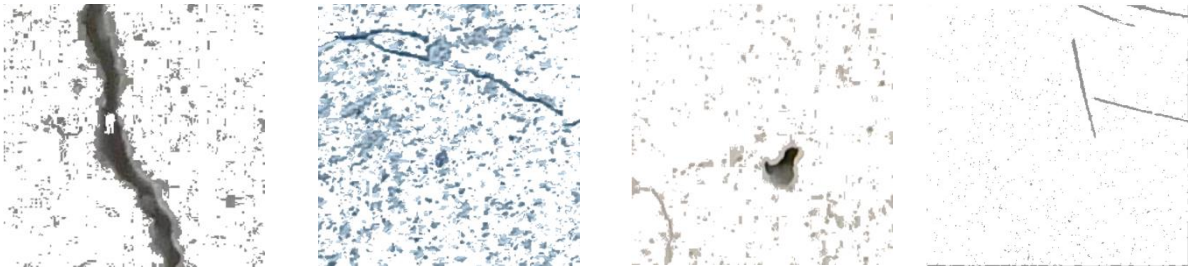
Процесс разрастания регионов не прекращается до тех пор, пока не останется ни одной точки изображения, которая могла бы быть присоединена к одному из уже выращенному региону.

Результат работы алгоритма приведен на рис. 2.

Из полученных результатов видно, что алгоритм справился с задачей сегментации. Стоит выделить результат сегментации на рис. 2 з) – это первый алгоритм из рассмотренных ранее, который смог точно выделить дефект без применения предварительной обработки. Однако, исходя из результатов на рис. 2 д) и рис. 2 е) можно сделать вывод о том, что алгоритм достаточно чувствителен к текстурным особенностям поверхности



а) *б)* *в)* *г)*



д) *е)* *ж)* *з)*

Рис 2. Результаты сегментации методом выращивания регионов. а-г) Изображения до сегментации, д-з) результаты после сегментации.

2. Измерение времени работы алгоритмов

В ходе исследования были выбраны 4 алгоритма сегментации, которые в дальнейшем будут использоваться в мобильном приложении. Было проведено измерение времени работы алгоритмов. В табл. 1 представлено время работы рассмотренных алгоритмов (метод к-средних [1], метод Оцу [1], метод водораздела и выращивания регионов), рассчитанное как среднее время его работы для каждого из четырех тестовых изображений.

Таким образом, самым неэффективным по времени и результатам сегментации оказался метод водораздела.

Сильное временное различие между остальными алгоритмами можно объяснить тем, что метод водораздела требует большого количества вычислений и содержит несколько вложенных циклов, что значительно увеличивает алгоритмическую сложность метода.

3. Разработка мобильного приложения для использования алгоритмов сегментации

Таблица 1

Время работы алгоритмов сегментации

Алгоритм	Время, с
Метод к-средних	4,49
Метод водораздела	17, 75
Метод Оцу	3,55
Метод выращивания регионов	6,76

Для применения рассмотренных алгоритмов на практике было разработано мобильное приложение на платформе Android с использованием языка Kotlin.

Приложение предоставляет пользователю следующие функции:

1. Выбрать изображения из галереи или самостоятельно сделать снимок.

2. Выбрать определенный алгоритм сегментации или применить сразу несколько из представленных.
3. Посмотреть результат сегментации изображения, а также получить информацию о времени выполнения того или иного алгоритма.

Однако алгоритмы сегментации, написанные на языке программирования Python не могут быть напрямую запущены из приложения, написанного на ЯП Kotlin. Для решения этой проблемы был использован SDK Chaquopy [5], позволяющий коду Python работать в Android-приложениях. Этот инструмент обладает полной интеграцией с системой сборки Gradle в Android Studio; позволяет использовать такие пакеты как, например, OpenCV, Matplotlib и пр.; а также предоставляет API, с помощью которого можно вызывать алгоритмы, написанные на Python, из Kotlin и наоборот.

Чтобы правильно настроить работу приложения с Python-кодом, нужно добавить в конфигурационный файл build.gradle [6] информацию об используемом плагине, пример представлен в Листинге 1.

Листинг 1

```
plugins {
    id 'com.android.application'
    id 'com.chaquo.python'
}
```

Кроме того, в конфигурационный файл также была добавлена информация о необходимой версии языка Python, о директории, в которой должны находиться алгоритмы сегментации, а также о библиотеках (напр. Matplotlib), необходимых для их работы.

Далее была написана функция на Kotlin, которая по имени модуля получает экземпляр объекта PyObject, предоставляющий возможность вызвать необходимую функцию по ее названию для того, чтобы выполнить необходимый метод сегментации. Пример функции предоставлен в Листинге 2.

Листинг 2

```
fun executeAlgorithm(path: String): String {
    val otsu = Python.getInstance().getModule("otsu_threshold")
    return otsu.callAttr("execute_otsu", path).toString()
}
```

Заключение

В результате проделанной работы были исследованы два дополнительных алгоритма сегментации: метод водораздела и выращивания регионов. Исходя из полученных результатов можно сделать вывод о том, что метод выращивания регионов эффективнее справляется с выделением дефекта поверхности. Также из-за особенности реализации метода водораздела, четкие границы сегментов препятствуют фильтрации итогового изображения таким образом, чтобы на нем остался только выделенный дефект.

Также в ходе работы были проведены измерения времени работы алгоритмов сегментации. Метод водоразделов также оказался неэффективным и с точки зрения времени выполнения: в 3-4 раза дольше остальных.

В данной работе было разработано мобильное приложение, с помощью которого была рассмотрена возможность внедрения алгоритмов, написанных на языке Python, в Android-приложения, написанных на Kotlin.

Литература

1. Шилова, И. Г. Исследование алгоритмов сегментации для выделения дефектов различных поверхностей / И. Г. Шилова, Е. В. Трофименко // Актуальные проблемы прикладной математики, информатики и механики: сб. тр. Междунар. науч.-техн. конф. (Воронеж, 12-14 декабря 2022 г.): электронный ресурс. – Воронеж, 2022. – С. 459-464

2. Гонсалес, Р. С. Цифровая обработка изображений / Р. С. Гонсалес, Р. Е. Вудс. – 3-е изд., исп. и доп. – Москва :Техносфера, 2012. – 1104 с.

3. Шапиро, Л. Компьютерное зрение / Л. Шапиро, Дж. Стокман; пер. с англ. – Москва: БИНОМ. Лаборатория знаний, 2006. – 752 с.

4. Цапаев, А. В. Методы сегментации изображений в задачах обнаружения дефектов поверхности / А. В. Цапаев, О. В. Кретинин // Компьютерная оптика. – 2012. – Т. 36. – № 3. – С. 448–452.

5. Chaquopy Documentation: https://chaquo.com/chaquopy/doc/current/ – (Дата обращения: 24.02.2023).

6. Kotlin docs https://kotlinlang.org/docs/home.html – (Дата обращения: 24.02.2023).

УДК 004.4

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ОЛИМПИАДНОГО ПРОГРАММИРОВАНИЯ С WEB-ИНТЕРФЕЙСОМ

Р. Р. Шимкин

Воронежский государственный университет

Введение

Целью работы является проектирование и создание информационной системы, представляющей собой web-приложение для организации олимпиад по программированию и автоматизации всех процессов их сопровождения, таких как разграничение условий доступа участников, безопасный и эффективный запуск прилагаемого ими кода и его тестирование, ведение списка результатов и т.д.

1. Структура и анализ функциональности

Приложение включает следующие модули:

1. «Олимпиады».
2. «Задания».
3. «Профиль».
4. «Администрирование».

Модуль «Олимпиады» доступен авторизованным пользователям с ролями «Участник» или «Организатор» и обеспечивает им следующие возможности:

1. Получение списка всех олимпиад, их фильтрация и сортировка по различным параметрам.
2. Получение списка задач и таблицы результатов выбранной олимпиады.
3. Отправка решения задач в виде загруженного файла или набранного во встроенном редакторе кода, получение информации о работоспособности и корректности кода.
4. Создание новых олимпиад, наполнение их задачами, настройка параметров их доступности.

Модуль «Задания» доступен авторизованным пользователям с ролями «Участник» или «Организатор», а также неавторизованным пользователям, и предлагает схожие с предыдущим модулем возможности для работы с отдельными задачами, не входящими в ограниченные по времени действия олимпиады.

Модуль «Профиль» доступен авторизованным пользователям и предоставляет функциональность для управления аккаунтом:

1. Регистрация и авторизация.
2. Изменение параметров входа и настроек учётной записи.
3. Получение и фильтрация информации о олимпиадах и задачах, в которых участвовал пользователь.
4. Доступ к списку созданных пользователем олимпиад.

Модуль «Администрирование» обеспечивает администратору сайта возможности по контролю за внутренними процессами компиляции и тестирования кода участников, ручному управлению параметрами учётных записей и олимпиад, администрированию базы данных, проверки журналов и логов приложения.

2. Технические требования

Приложение должно обеспечивать приемлемое время обработки запросов и тестирования пользовательского кода при одновременной работе с сайтом многих пользователей.

При запуске скомпилированных программ должна быть обеспечена защита системы от случайного или намеренного вредного воздействия пользовательских программ на внутренние процессы сервера, от влияния на результаты выполнения кода других пользователей. То есть должны быть реализованы изоляция запущенных программ и ограничения прав доступа вне пользовательских файлов.

Система должна быть отказоустойчивой, необходимо журналирование и логирование всех процессов обработки web-запросов, работы с кодом и изменений данных.

Архитектура приложения должна быть гибкой, чтобы имелись возможности простого расширения функциональности (добавление новых видов задач, добавление новых языков программирования).

Работа с базой данных должна быть организована с учётом возможности сменить вид используемой СУБД, быстро внести изменения в схему данных.

Интерфейс сайта должен быть максимально понятным и оперативно реагировать на действия пользователя и события на сервере. Отправка кода и получение результата его работы должны выполняться без асинхронно без обновления страницы.

Пользователям должен быть обеспечен полноценный доступ к функциональности сайта и корректное визуальное отображение на всех современных браузеров, включая мобильные.

Приложение должно допускать быстрое развертывание на сервере и быстрый перезапуск при изменении параметров среды. Приоритетная ОС – Linux.

3. Средства реализации

Для реализации серверной части системы был выбран Python 3, так как это популярный кроссплатформенный язык программирования с развитым сообществом разработчиков и богатой встроенной библиотекой, обеспечивающей все необходимые возможности взаимодействия с операционной системой, файловой системой и компиляторами других языков.

В качестве web-фреймворка используется Django, как самое мощное решение для разработки на Python, предоставляющее гибкую ORM-систему для удобной работы с различными типами баз данных, а также преимущества модульной архитектуры приложения. Фреймворк реализует MVT-архитектуру (модель, представление, шаблон).

Для хранения данных выбрана СУБД MongoDb, реализующая документную модель. Её преимуществами является потенциально высокая скорость работы с данными, гибкое изменение схемы данных без необходимости полного обновления базы, возможность хранения данных в виде массивов и вложенных объектов.

На стороне клиента используются HTML 5, CSS, JavaScript ES6. Для организации удобного редактора кода в браузере используется библиотека CodeMirror.

Разворачивать приложение планируется на операционной системе Linux, так как она удобнее для серверов и предоставляет больше механизмом разграничения доступа и изоляции запускаемых процессов (chroot, контейнеризация Docker, kvm, seccomp и другие).

База данных содержит коллекции «Profile», «User», «Olympiad», «Task», «TestData», «TaskResult», «OlympiadResult». Наследуемые сущности хранятся в общих таблицах.

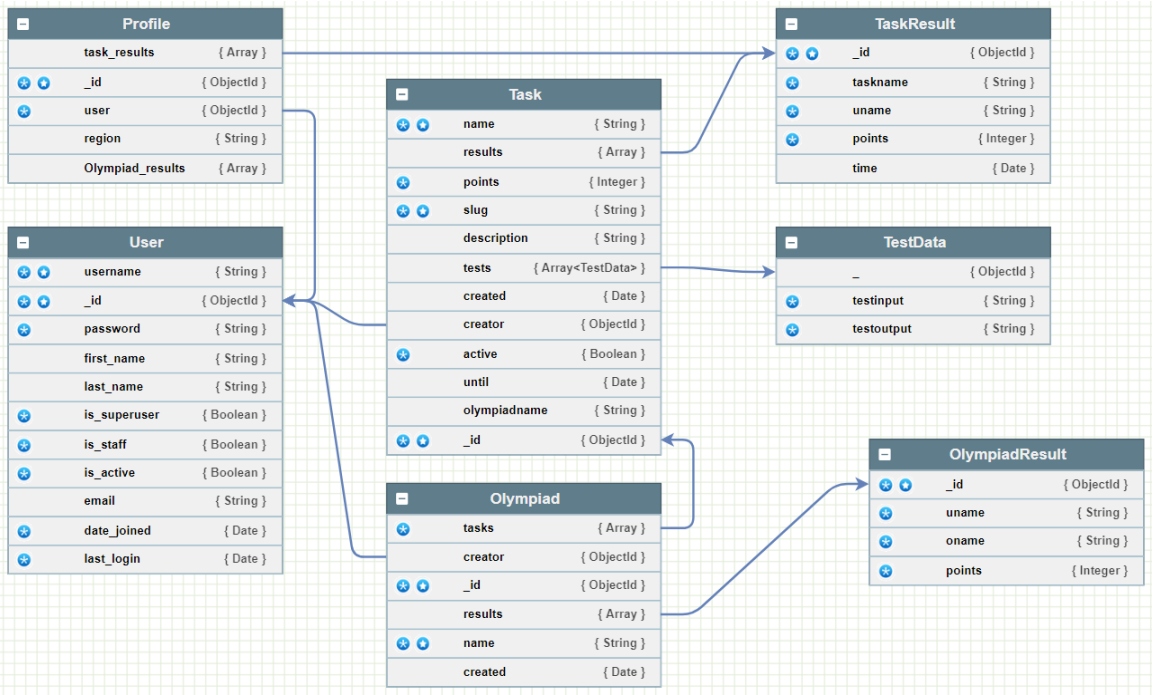


Рис. 1. Схема базы данных

Заключение

В результате работы создано web-приложение, отвечающее всем требованиям постановки задачи. Оно предоставляет функциональность работы с олимпиадами по программированию, обеспечивает безопасность сервера и пользователей, предполагает возможности настройки и последующего расширения. Информационная система рассчитана на малую и среднюю нагрузку в зависимости от используемых серверных мощностей.

Литература

1. Lutz M. Learning Python 5th Edition: – O'Reilly Media Inc., 2013. – 1542 с.
2. Дронов В.А. Django 3.0. Практика создания веб-сайтов на Python. — СПб.: БХВ-Петербург, 2021. — 704 с.
3. Брэдшоу Ш. MongoDB: полное руководство. Мощная и масштабируемая система управления базами данных / Ш. Брэдшоу, Й. Брэзил, К. Ходоров /пер. с англ. Д. А. Беликова – М.: ДМК Пресс, 2020. – 540 с.
4. Django documentation. – Режим доступа: https:// docs.djangoproject.com (Дата обращения: 20.03.2023).

УДК 004.774

РАЗРАБОТКА JAVASCRIPT-БИБЛИОТЕКИ ДЛЯ ПОСТРОЕНИЯ ДИАГРАММ

А. Ю. Ширнин

Воронежский государственный университет

Введение

С ростом спроса на визуальное представление данных JavaScript-библиотеки для создания и отображения диаграмм стали важным компонентом многих веб-приложений. Однако большинство существующих библиотек могут удовлетворять не всем требованиям, что приводит к необходимости создания собственных библиотек. С этой целью в данной статье анализируется процесс разработки пользовательской библиотеки JavaScript для отрисовки диаграмм.

1. Основные этапы разработки библиотеки

1. Требования. Первым шагом является определение требований к библиотеке, таких как виды диаграмм, которые будут создаваться, необходимый уровень их кастомизации и любые специфические функции.
2. Технологический стек. На основе выявленных требований необходимо выбрать подходящий технологический стек, включая фреймворки и библиотеки в виде зависимостей, для разработки собственной библиотеки.
3. Структура. Определение базовой структуры библиотеки, включая структуру каталогов, организацию файлов и основной файл JavaScript.
4. Разработка. По выбранной функциональности разработать библиотеку, включая возможность создания различных типов диаграмм, инструменты рисования и взаимодействие с пользователем.
5. Тестирование и отладка. Важный этап создания библиотеки — тестирование функциональности и отладка всех возникающих проблем. Данный шаг может включать создание образцов диаграмм и тестирование различных сценариев использования.
6. Доработка и оптимизация. После тестирования необходимо доработать и оптимизировать работу библиотеки, чтобы она могла обеспечивать оптимальный пользовательский опыт.
7. Документация. Хорошо описанная функциональность библиотеки и API помогает понять разработчикам ее назначение и гарантирует эффективное использование, облегчая разработчикам интеграцию библиотеки в свои проекты.
8. Развертывание. После развертывания библиотеки в публичном репозитории или реестре, таком как NPM или GitHub, она становится доступной для широкой аудитории, которая может легко установить и использовать ее в своих приложениях.

Эти этапы являются итеративными и требуют постоянной обратной связи и улучшения на протяжении всего процесса разработки, чтобы убедиться, что библиотека соответствует требованиям и функционирует оптимально.

2. Анализ средств разработки и стека

Выбор правильного стека технологий и инструментов имеет решающее значение для успеха библиотеки. Ниже описаны некоторые из наиболее распространенных инструментов и технологий, используемых при разработке библиотек.

2.1. Сборщики проектов

В настоящее время самыми популярными сборщиками веб-проектов являются Rollup и Webpack. Они схожи между собой по своей основной функциональности, но имеют некоторые ключевые различия. Оба сборщика тесно связаны с Node.js и работают с различными видами файлов, собирают каждый и сокращают код в файлы по их типам. Вследствие чего можно оптимизировать кодовую базу, разбивая ее на модули, повысить производительность приложения из-за объединения модулей в один файл.

В свою очередь Rollup фокусируется на tree-shaking, технике удаления неиспользуемого кода из конечного пакета, что приводит к уменьшению размера и ускорению загрузки библиотек. А Webpack использует графовый подход для анализа зависимостей между модулями, это мощный инструмент для создания сложных веб-приложений с множеством точек входа, плагинов и загрузчиков.

2.2. Менеджеры зависимостей JavaScript пакетов

Менеджеры пакетов используются для управления зависимостями библиотеки и обеспечения простоты ее установки и использования. На данный момент есть два популярных менеджера зависимостей — npm и Yarn.

npm — это менеджер пакетов по умолчанию для Node.js, и он широко используется в экосистеме JavaScript. npm имеет обширную библиотеку пакетов, что делает его отличным выбором для разработчиков, которые хотят создавать сложные приложения.

Yarn — это относительно новый менеджер пакетов, созданный компанией Facebook. Он был разработан для устранения некоторых ограничений npm.

Одним из основных различий между npm и Yarn является скорость установки. Yarn использует параллельную установку, что позволяет ему устанавливать пакеты быстрее, чем npm. Yarn также имеет систему кэширования, которая позволяет повторно использовать уже установленные пакеты, что еще больше сокращает время установки.

2.3. Анализ кода и тестирование

Чтобы убедиться, что библиотека хорошо спроектирована и не содержит ошибок, можно использовать линтеры и фреймворки тестирования.

Линтеры — это инструменты, которые анализируют код на предмет потенциальных ошибок и нарушений стиля. Они полезны для поддержания согласованности в проектах, выявления ошибок на ранних стадиях и улучшения общего качества кода. Популярные линтеры для JavaScript являются ESLint, JSLint и JSHint.

Фреймворки для тестирования — это инструменты для автоматизации процесса тестирования и обеспечения того, чтобы изменения кода не приводили к новым ошибкам или регрессиям. Они позволяют разработчикам писать наборы тестов, которые автоматически проверяют поведение кода, гарантируя, что код функционирует так, как задумано. Популярные фреймворки для тестирования JavaScript: Jest, Mocha и Jasmine.

2.4. Инструменты работы с canvas

Canvas — это HTML-элемент для отрисовки графики через скрипты. Он позволяет разработчикам рисовать графику и анимацию непосредственно на веб-странице с помощью JavaScript.

Для рисования на элементе canvas разработчики используют API CanvasRenderingContext2D, который является API JavaScript и предоставляет набор методов и свойств для рисования фигур, линий, текста и изображений на холсте.

В целом, элемент canvas и API CanvasRenderingContext2D обеспечивают мощную и гибкую платформу для рисования графики и анимации в Интернете с помощью JavaScript. С помощью этих инструментов разработчики могут создавать сложные и интерактивные визуальные эффекты для пользователей.

В дополнение к стандартному API существует несколько сторонних библиотек и фреймворков для рисования на элементе canvas в JavaScript. К наиболее популярным библиотекам и фреймворкам относятся Fabric.js, Paper.js и Konva.js. Но, как правило, лучше не создавать внешние зависимости в JS-библиотеке, потому что это может сделать ее менее портативной, менее безопасной и менее надежной.

3. Лучшие практики

После того, как выбран правильный стек для библиотеки JavaScript, следует знать несколько лучших практик при разработке библиотеки.

Важно помнить принцип DRY (Don’t repeat yourself), что означает "Не повторяй себя". Этот принцип побуждает разработчиков избегать дублирования кода и создавать многократно используемые компоненты, которые можно использовать во всей библиотеке. Следуя этому принципу, можно упростить процесс разработки и снизить риск возникновения ошибок или багов.

Необходимо писать четкую и лаконичную документацию, которая объясняет, как использовать функциональность библиотеки, а также любые требования или зависимости, которые могут потребоваться. Хорошая документация поможет обеспечить широкое распространение библиотеки и ее использование другими разработчиками.

Инструменты контроля версий, такие как Git, позволяют легко управлять изменениями в кодовой базе с течением времени. Используя контроль версий, можно без труда отслеживать изменения, сотрудничать с другими разработчиками и следить за актуальностью библиотеки.

Производительность является ключевым фактором для любой библиотеки JavaScript. Важно используйте такие инструменты, как Webpack, чтобы оптимизировать библиотеку для производительности и убедиться, что она корректно работает на различных устройствах и браузерах.

4. Публикация библиотеки

После разработки библиотеки JavaScript, следующим шагом будет ее развертывание и публикация в менеджере пакетов, например, npm. Для этого необходимо зарегистрироваться на сайте менеджера пакетов и с помощью нескольких команд в терминале проекта опубликовать его.

Публикация библиотеки имеет ряд преимуществ:

1. Охват широкой аудитории разработчиков, которые ищут решения своих проблем.
2. Простая установка и интеграция в свой проект другими пользователями.
3. npm предоставляет инструменты управления версиями и обновлениями, которые

позволяют легко управлять изменениями в коде с течением времени.

Таким образом, публикация библиотеки поможет поднять аудиторию разработчиков, упростить установку и использование, управлять обновлениями, получить поддержку сообщества и признание работы.

Заключение

Разработка библиотеки JavaScript для отрисовки диаграмм на холсте включает в себя несколько этапов и требует тщательного планирования и реализации. Используя правильные инструменты разработки и следуя лучшим практикам, можно создать высококачественную библиотеку, которая удовлетворит потребности пользователей и внесет вклад в развитие сообщества разработчиков.

В результате работы был проанализирован стек разработки, включая такие инструменты, как Webpack и Rollup, важность линтеров и фреймворков для тестирования, а также рассмотрен HTML-элемент canvas и JavaScript API для рисования на нем. Выделены лучшие практики разработки JavaScript-библиотек, такие как версионирование, избегание внешних зависимостей и тщательное документирование.

Литература

1. Flanagan, D. JavaScript: The Definitive Guide / D. Flanagan – 7-е изд., перераб. и доп. – Sebastopol: O'Reilly Media, Inc., 2020. – 1096 с.

2. Elliott, E. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries / E. Elliott – 1-е изд., Sebastopol: O'Reilly Media, Inc., 2014. – 254 с.

3. MDN Web Docs. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/JavaScript. – (Дата обращения: 15.03.2023).

4. The npm website. – Режим доступа: https://www.npmjs.com/. – (Дата обращения: 12.03.2023).

УДК 519.15

ЗАДАЧА НЕЧЕТКОГО ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ С ЧЕТКОЙ ЦЕЛЕВОЙ ФУНКЦИЕЙ

М. М. Шишов

Воронежский государственный университет

Аннотация. В статье рассматривается задача нечеткого линейного программирования с четкой целевой функцией и подходы к её решению.

Ключевые слова: нечеткое линейное программирование, агрегирование, α-срезы.

Введение

В последнее время проблема агрегирования активно обсуждается в научной литературе, ввиду того, что лежит в основе большинства процедур принятия решений, используется в нейросетевых технологиях, которые в настоящее время набирают всё большую популярность. Агрегирование – переход от векторной оценки размерности n к векторной оценке размерности m, при этом $m<n$ [5]. В данной статье будет рассмотрена задача нечеткого линейного программирования с четкой целевой функцией, а также два подхода к решению проблемы агрегирования.

Подходы к решению проблемы агрегирования

Модель не является симметричной в том случае, если: целевая функция четкая, а ограничения частично или полностью нечеткие. Роли целевой функции и ограничений разные. Проблема здесь заключается в пересчете целевой функции при агрегировании с нормализованными ограничениями. Чтобы решить эту проблему, возможно применить два подхода: определить нечеткое множество “решений”, либо определить четкое множество “максимизирующих решений” путем агрегирования целевой функции после необходимых преобразований с ограничениями.

Вначале рассмотрим первый подход: определение нечеткого множества “решений”.

1. Определение нечеткого множества “решений”

Пусть $R_\alpha = \{x : x \in X, \mu_R(x) \geq \alpha\}$ - множество уровня α (α-срез) в пространстве решений и $N(\alpha) = \{x : x \in R_\alpha, f(x) = \sup\limits_{x' \in R_\alpha} f(x')\}$ - множество оптимальных решений для каждого α - уровневого множества.

Тогда нечеткое множество “решений” определяется следующей функцией принадлежности:

$$\mu_{opt}(x) = \begin{cases} \sup\limits_{x \in N(\alpha)} \alpha, \text{ если } x \in \bigcup\limits_{\alpha > 0} N(\alpha) \\ 0, \text{ иначе} \end{cases}$$

В таком случае нечеткое множество “оптимальных значений целевой функции” имеет следующую функцию принадлежности

$$\mu_f(r) = \begin{cases} \sup\limits_{x \in f^{-1}(r)} \mu_{opt}(x), \text{ если } r \in R_1 \text{ и } f^{-1}(r) \neq \varnothing \\ 0, \text{ иначе} \end{cases}$$

где, $f(x)$ -целевая функция со значениями r.

В случае линейного программирования определение значений r и $\mu_{opt}(x)$ происходит с помощью параметрического программирования.

Для каждого α необходимо решить следующую задачу линейного программирования:

$$f(x) \to \max$$
$$\mu_i(x) \geq \alpha, i = 1, ..., m$$
$$x \in X$$

В результате получим нечеткое множество. Соответственно, лицо принимающее решение (ЛПР) должно будет решить, какая пара $(r, \mu_f(r))$ будет наилучшей для принятия r в качестве четкого оптимального решения.

2. Определение четкого множества максимизирующих решений

Введем следующее определение.

Пусть $f : X \to R^1$ - целевая функция, R - нечеткое множество (пространство решений), $S(R)$ - носитель этого множества. Множество максимимзирующих решений этого нечеткого множества $MR(f)$, определяется функцией принадлежности [2]:

$$\mu_{MR}(x) = \begin{cases} 0, \text{ если } f(x) \leq \inf\limits_{S(R)} f, \\ \dfrac{f(x) - \inf\limits_{S(R)} f}{\sup\limits_{S(R)} f - \inf\limits_{S(R)} f}, \text{ если } \inf\limits_{S(R)} f < f(x) < \sup\limits_{S(R)} f, \\ 1, \text{ если } \sup\limits_{S(R)} f \leq f(x). \end{cases}$$

Для вычисления максимизирующего решения x_0 может быть использовано пересечение множества максимизирующих решений и нечеткого множества “решений”. Кроме того, максимизирующее решение x_0 может быть принято в качестве решения с максимальной степенью принадлежности этому нечеткому множеству.

Также рассмотрим еще одно определение.

Пусть $f : X \to R$ -целевая функция, R нечеткое множество (пространство решений), $S(R)$ носитель R, R_1 - α -срез для R при $\alpha = 1$. Функция принадлежности цели, заданная пространством решений R, определяется следующим образом:

$$\mu_G(x)=\begin{cases}0,\ \textit{если}\ f(x)\le \sup\limits_{R_1} f,\\ \dfrac{f(x)-\sup\limits_{R_1} f}{\sup\limits_{S(R)} f-\sup\limits_{R_1} f},\ \textit{если}\ \sup\limits_{R_1} f<f(x)<\sup\limits_{S(R)} f,\\ 1,\ \textit{если}\ \sup\limits_{S(R)} f\le f(x).\end{cases}$$

Соответствующая функция принадлежности в пространстве функций имеет следующий вид

$$\mu_G(r)=\begin{cases}\sup\limits_{x\in f^{-1}(r)} \mu_G(x),\ \textit{если}\ r\in R\ \textit{и}\ f^{-1}(r)\neq\varnothing\\ 0,\ \textit{иначе}\end{cases}$$

Заключение

Таким образом, в данной работе была рассмотрена задача нечеткого линейного программирования с четкой целевой функцией. Также, сделан обзор на два подхода, которые помогают решить проблему пересчета целевой функции при агрегировании с нормализованными ограничениями.

Работа выполнена под руководством профессора, д-ра техн. наук, заведующего кафедрой вычислительной математики и прикладных информационных технологий ВГУ, Леденёвой Татьяны Михайловны.

Литература

1. Леденева, Т.М. Агрегирование информации в оценочных системах / Т.М. Леденева, С.Л. Подвальный // Вестн. Воронеж. гос. ун-та. Серия: Системный анализ и информационные технологии. – 2016. – №4. – С. 155-164.

2. Юдин Д. Б. Задачи и методы линейного программирования. Математические основы и практические задачи /Д. Б. Юдин, Е. Г. Гольштейн. – Москва : Либроком, 2010. – 322с.

3. Беллман Р. Э. Принятие решений в расплывчатых условиях / Р. Э. Беллман, Л. А. Заде. – М. : Мир, 1976. – 357 с.

4. Орловский С. А. Проблема принятия решений при исходной нечеткой информации / С. А. Орловский. – М. : Наука, 1981. – 206 с.

5. Аристова Е.М. Упрощение задачи линейного многокритериальной оптимизации с помощью метода агрегирования целевых функций / Е.М. Аристова// Вестник ВГУ, Серия: Системный анализ и информационные технологии. – Воронеж, 2012. – №2. – С. 11-17.

УДК 004

ПРОСТРАНСТВЕННОЕ ИНДЕКСИРОВАНИЕ ДЛЯ ПОДВИЖНОЙ ГЕОМЕТРИИ В ОПЕРАТИВНОЙ ПАМЯТИ

В. Т. Штанько

Воронежский государственный университет

Введение

Одним из основополагающих факторов, влияющих на быстродействие геоинформационной системы – системы, предназначенной для эффективной обработки информации о географических объектах и их визуализации, – является асимптотическая сложность стратегий поиска и обновления информации о местоположении и границах географических объектов, хранимых в базе данных этой системы. Общее название структур данных, обеспечивающих наименьшее время выполнения пространственных запросов, с которыми так часто приходится иметь дело каждой ГИС, носит название пространственного индекса. На сегодняшний день существует множество различных структур данных, предназначенных для пространственного индексирования, однако для большинства из них справедлив тот факт, что высокая скорость выполнения поисковых пространственных запросов не является гарантией того, что обновление и добавление данных будет так же высокопроизводительно; также к настоящему времени было проведено достаточно мало исследований свойств пространственных индексов для геометрии, представляемой отрезками, полигонами и другими неточечными объектами и требующей очень частого обновления.

В рамках данной статьи будут рассмотрены с точки зрения эффективности поиска и обновления данных наиболее распространенные структуры данных и алгоритмы, используемые для индексирования подвижной геометрии; также будет рассмотрена структура данных, основанная на двухсторонне прошитом бинарном дереве поиска с центрированным обходом, и будут сделаны оценки сложности алгоритмов обновления и поиска данных для нее.

1. Проблематика пространственного индексирования

1.1. Пример приложения: предупреждение и урегулирование ЧС

Рассмотрим в качестве примера приложение, необходимое для операторов отдела предупреждения и урегулирования чрезвычайных ситуаций. Предполагается, что центральный сервер этого отдела постоянно отслеживает координаты большого множества позиционных данных, таких как данные о мобильных устройствах, автомобилях, погодных явлениях, используя при этом различные источники. Хранимая на сервере информация о местоположении объектов обновляется с высокой частотой: одно обновление в секунду для каждого объекта. Так как определяемые местоположения объектов заведомо содержат некоторую неточность, порожденную погрешностями GPS-позиционирования или определения местоположения через вышки сотовой связи, то данные, изначально представляющие координаты объектов-точек, также обладают неточностью, вследствие чего в приложении они представлены как геометрические объекты с ненулевой площадью.

Каждый оператор – работник отдела предупреждения чрезвычайных ситуаций – взаимодействует с сервером через клиентскую программу. На экране монитора своего рабочего компьютера он видит окно, на котором визуализируется участок карты с данными, поступающими от сервера. Важно, чтобы предоставляемая сервером информация была настолько актуальной, насколько возможно, то есть чтобы на карте, используемой оператором, для каждого рассматриваемого им участка, данные о находящихся на нем в текущий момент времени объектов были как можно более достоверны; также необходимо обеспечить возможность быстро переходить от одного участка карты приложения к другому, увеличивая или уменьшая масштаб.

В терминологии пространственных запросов термин «оконный запрос» как раз соответствует запросу, используемого оператором: найти все объекты, либо находящиеся внутри заданного осепараллельного прямоугольника (окна), либо же пересекающие его границы.

1.2. Основные свойства пространственных запросов

Пространственный индекс — это структура данных, предназначенная для хранения объектов, которые представлены точками или иными видами геометрии (в двух- и трехмерных пространствах), а также для эффективного их поиска посредством так называемых пространственных запросов.

Пространственное индексирование и варианты его применения представляют собой достаточно обширное поле для исследований. В данной же статье учитываются пространственные запросы, удовлетворяющие следующим свойствам:

1. Двухмерные данные. Все рассмотренные в данной статье структуры данных предполагают возможность расширения до любой размерности, однако на практике использование пространственных индексов ограничивается двух- и трехмерными пространствами.

2. Осепараллельные прямоугольники. Любой геометрический объект может быть аппроксимирован своим минимальным ограничивающим прямоугольником, что является уже традиционной методикой первичной обработки исходных объектов реального мира – такое упрощение значительно сокращает расходы на хранение и обработку информации об объекте.

3. Оперативная память. Базы данных, использующие пространственные индексы, как правило хранятся во внешней памяти, где операции чтения данных с диска и записи данных на диск являются основными факторами, ограничивающими производительность работы с данными. Однако многие задачи, среди которых – и приведенный ранее пример, вполне могут быть решены с использованием только оперативной памяти, соответственно рассматриваемые в дальнейшем структуры данных предусматривают хранение их самих и индексируемых ими объектов непосредственно во внутренней памяти ЭВМ.

4. Оконные запросы. Множество запросов, применимых и применяемых к пространственным данным достаточно велико и включает в себя такие типы операции, как запросы к данным на совпадение отдельных координат, диапазонные запросы, поиск ближайших соседних объектов, определение местонахождения объекта (то есть для заданной точки запрашиваются сведения о содержащих ее фигурах или областях; синонимичный термин – секущий запрос). На текущем этапе мы сделаем акцент именно на диапазонных запросах, которые в случае двухмерных данных также носят название *оконных запросов*.

2. Структуры данных, используемые для пространственного индексирования

Существует несколько различных категорий структур данных, используемых для хранения информации о пространственных объектах. В качестве наиболее базовых категорий

можно выделить такие, как *хеш-таблицы* и *древовидные структуры*. Древовидные структуры, очевидно, обладают рядом недостатков, не свойственных хеш-таблицам: во-первых, они утрачивают свойство сбалансированности и ветви целого ряда распространенных древовидных структур склоны к разрастанию, в то время как другие ветви остаются короткими (примером такой структуры является *дерево квадрантов*); во-вторых, при использовании различных модификаций деревьев всегда приходится жертвовать скоростью обновления данных (в том числе вставки и удаления). Однако важным преимуществом таких структур является возможность более адекватной оценки пространственной близости двух различных геометрических объектов, чем та, которую дает использование хеш-таблиц. Так как в данной статьей делается особый акцент именно на оконные запросы, для которых более оправдано использование древовидных структур, то именно некоторые модификации последних мы и рассмотрим в данном разделе.

2.1. Дерево квадрантов

Наиболее распространенным типом дерева-квадрантов [6] являются такие, для которых соответствующее разбиение пространства не зависит от распределения индексируемых точек. В каждой промежуточной вершине дерева хранятся координаты середины соответствующего квадранта разбиения. В случае размерности k каждая нелистовая вершина ссылается на 2^k дочерних вершин, каждая из которых соответствует одному из 2^k одинаковых по размерам квадрантов. Необходимость в разбиении узла на дочерние определяется заданным максимальным числом точек, которые могут хранится в одном узле.

Дерево квадрантов адаптируемо под хранение различных геометрических объектов точек. Однако при хранении неточечных объектов важно определиться со стратегией размещения объектов, пересекающих сразу несколько квадрантов. Одним из возможных решений данной проблемы является размещение объектов не только в листовые, но и в промежуточные узлы дерева. Другие способы разрешения этой проблемы связаны с разбиением индексируемого объекта на части, каждая из которых лежит в своем листовом узле, или реализацией ссылок на данный объект из перекрываемых им квадрантов. Но при таком решении вычислительная сложность операций, связанных с обновлением дерева, сильно возрастает.

Сложности вставки и поиска точечных объектов в дереве квадрантов имеют достаточно позитивную оценку – $O(\log n)$, однако, как уже было сказано выше, необходимость в операциях над объектами с ненулевой площадью сильно ухудшает данную оценку.

2.2. R-дерево

Предложенное А. Гуттманом [3] в 1984г. R-дерево представляет собой структуру данных, подобную B-дереву, зарекомендовавшего себя наиболее эффективным для индексации линейно упорядоченных множеств. R-дерево, изначально описанное как структура данных для двухмерных объектов. представляет из себя разбиение индексируемого пространства, как и в случае деревьев квадрантов, на иерархически вложенные прямоугольники; при этом, однако, в некоторых модификациях R-деревьев допустимо и пересечение прямоугольников друг с другом. Создание такой системы прямоугольников обусловлено стремлением сосредоточить близко расположенные объекты в одном прямоугольнике. Как правило, для вставки очередного объекта в R-дерево отбирается такой из существующих прямоугольников дерева, для которого потребуется наименьшее расширение его площади. При поиске же объектов в дереве используется типичная для древовидных структур данных стратегия: начиная от корня дерева и передвигаясь к листовым вершинам, отсеиваются такие узлы, для которых координаты соответствующих им прямоугольников заведомо не удовлетворяют условию поиска.

Основными же недостатками R-дерева являются сильная зависимость качества его построения от порядка, в котором добавляются объекты, а также то, что ограничивающие прямоугольники, соответствующие узлам разных уровней, могут перекрываться, что значительно ухудшает асимптотику работы основных алгоритмов для данной структуры. Так, для алгоритма поиска не существует более положительной худшей оценки, чем $O(n)$.

2.3. Приоритетные R-деревья

Приоритетное R-дерево (Priority R-tree), являющееся одной из модификаций R-дерева и алгоритмы для его построения и поиска объектов в нем были впервые предложены в 2004 г. [5]. Приоритетное R-дерево является по сути гибридом k-d дерева и классического R-дерева в том смысле, что в структура его также основывается поддержании системы иерархически вложенных прямоугольников, наименьшие из которые представляют собой минимальные ограничивающие прямоугольники для индексируемых объектов. Особенностью данной древовидной структуры является то, что в каждой ее ветви хранятся четыре дополнительных узла, в которые помещены крайние значения по каждому из измерений. Для ответа на обычный оконный запрос, перед тем как переходить от текущего узла к дочерним, в алгоритме поиска проверяется, перекрываются ли между собой области его дополнительных узлов (узлов приоритетов). Дочерние узлы просматриваются с параллельной проверкой того, превышает ли наименьшее значение первой координаты-измерения окна запроса соответствующее значение проверяемого дочернего узла.

Дерево приоритетов, в отличие от R-дерева, обладают асимптотически более оптимальной оценкой для худшего случая алгоритма поиска. Для оконного запроса эта оценка представляется как $O(\sqrt{(\frac{n}{B})} + \frac{T}{B})$, где n – количество двухмерных прямоугольников, хранящихся в дереве, B – размер дискового блока, а T – размер вывода.

Несмотря на то, что на практике приоритетные R-деревья превосходят различные модификации R-деревьев, включая R*-дерево и гильбертово R-дерево, в теории не доказано, что асимптотически они являются более оптимальными для худшего случая.

3. Предлагаемая структура данных

3.1. Бинарное дерево, прошитое для двухстороннего центрированного обхода

Прежде чем перейти к описанию структуры, оптимизирующей запросы в двухмерном пространстве, рассмотрим структуру данных, близкую по своей организации к дереву отрезков, однако обладающую рядом отличных особенностей, а именно – двухсторонне прошитое бинарное дерево поиска с центрированным обходом, которое обладает следующими свойствами [2].

1. Каждый узел v дерева T представлен как v ([a, b], L), где [a, b] - это диапазон, который соответствует узлу v, и L - это список сегментов, включающих в себя диапазон [a, b]: т.е. если [c, d] ∈ L, то [a, b] ⊆ [c, d].

2. Пусть $v([a,b],L) \neq w([c,d],N)$. Тогда $[a,b] \cap [c,d] = \begin{bmatrix} b, c = b \\ d, a = d \\ \infty, c \neq b, a \neq d \end{bmatrix}$

Иными словами, диапазоны могут перекрываться либо только своими концами, либо не перекрываются совсем.

3. Пусть $v([a,b],L)$ предшествует узлу $w([c,d],N)$ в последовательности узлов, посещаемых при inorder-обходе (центрированном обходе); тогда $b \le d$.

4. Содержит особый узел, называемый *dummy-узлом* и обозначаемый D, которому соответствует пустой диапазон и пустой список.

5. Пусть $v([a,b],L)$ и $z([e,f],K)$ - это первый и последний узлы, посещенные при центрированном обходе соответственно; тогда $In\,\mathrm{Pr}\,ed(v) = InSucc(z) = D$, и диапазон $[p,q]$ любого узла дерева содержится в $[a,f]$, то есть $[p,q] \subseteq [a,f]$.

Здесь функции $In\,\mathrm{Pr}\,ed()$ и $InSucc()$ соответственно возвращают предыдущий и последующий узлы при центрированном обходе относительно текущего.

Пример такого дерева показан на рис.1.

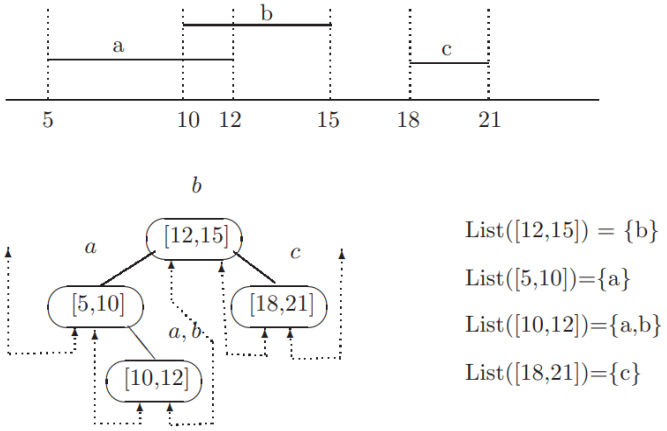


Рис.1. ДПБДПЦО первого порядка

Замечание 1. Свободно свисающие нити на самом деле адресуют головной (dummy) узел, который не показан на рисунке. Заметим, что на рис.3 соответствующий каждому узлу список хранит имена всех одномерных сегментов, которые либо содержатся в узле, либо перекрываются им слева или справа.

Замечание 2. Пусть есть два пересекающихся диапазона (сегмента) S и T. Объединение $S \cup T$ этих диапазонов (пусть это будет диапазон $[a,b]$) представляет из себя следующие три части (также являющиеся диапазонами):

- $C(S,T) = S \cap T = [p,q]$ – пересечение диапазонов S и T;
- $L(S,T) = [a,p]$, если $a \ne p$, иначе – пустое множество;
- $R(S,T) = [q,b]$, если $q \ne b$, иначе – пустое множество;

Замечание 3. Пусть $S = [p,q], T = [m,n]$ – это два пересекающихся отрезка, а $list(S), list(T)$ – это соответствующие им списки (то есть списки сегментов, которые включают в себя данные отрезки S или T соответственно). Тогда списки, соответствующие сегментам $L(S,T), C(S,T), R(S,T)$ определяются следующим образом:

$$list(C(S,T)) = list(S) \cup list(T)$$

$$list(L(S,T)) = \begin{bmatrix} list(S),\ p < m \\ list(T),\ p > m \\ \text{O}, иначе \end{bmatrix}$$

$$list(R(S,T)) = \begin{bmatrix} list(S), n < q \\ list(T), n > q \\ \text{O}, иначе \end{bmatrix}$$

Вставка в дерево

Сложность алгоритма вставки: $\text{O}(\log n + k)$, где k - количество узлов, в которые происходит вставка отрезка.

Чтобы вставить отрезок S в дерево, необходимо, как и по обычному бинарному дереву поиска, пройти от корня вниз и найти узел, диапазон которого перекрывается с заданным отрезком S. Пусть найден некоторый такой узел v, с соответствующим ему диапазоном R. И $S \cup R$ разбивается на части $L(S,T), C(S,T), R(S,T)$, как описано в замечании 2. Теперь диапазон, описываемый узлом v, заменяется на новый: $C(S,R) = S \cap R$, а в список $list(v)$ добавится идентификатор-ссылка на отрезок S. Далее происходит вставка диапазона $L(S,R)$ (если он не пуст) в непосредственного inorder-предшественника узла v. Аналогично, если $R(S,R)$ не пуст, то производится его вставка в inorder-последователя узла v. Однако в этих двух случаях список вставляемых сегментов может меняться. То есть изначально список вставляемых в дерево сегментов содержит только S, а в дальнейшем может измениться, в соответствии с замечанием 3. В какой-то момент процесс вставки может привести к нулевому указателю на следующий узел. В таком случае, еще не рассмотренный подотрезок вставляемого отрезка принимается за отрезок для нового узла дерево (то есть его нужно будет создать на текущей итерации); помимо него к этому новому узлу будут добавлены соответствующие указатели на inorder-соседей, а также список имен сегментов, которые требуется вставить в данный момент. Также, учитывая то, что мы строим сбалансированное дерево, вставка нового узла может потребовать процедуры вращения поддерева, которая сбалансирует увеличившееся в длину дерево. При реализации алгоритма вставки можно основываться на вращениях, применяемых к базовым AVL-деревьям.

Удаление из дерева

Пусть требуется удалить сегмент $S = [p,q]$. Для этого в дереве находим узел v с диапазоном $[p,q^*]$, где q* – это значение меньшее или равное q. Если такой узел не найден, значит отрезка S в дереве нет, и процесс удаления можно завершить. В противном случае сегмент S требуется удалить из списка $list(v)$. Если после этого список $list(v)$ оказался пустым, то соответствующий ему диапазон $[p,q^*]$ можно рассматривать как пустой, и узел v сливается с его предшественником $In\,\text{Pr}\,ed(v)$. В этом случае сливаются узлы с разных уровней - потребуется сделать некоторые вращения, так необходимо поддерживать сбалансированность дерева. Если же $list(v)$ оказался непустым после удаления сегмента S, то нужно проверить, совпадает ли $list(v)$ с $list(In\,\text{Pr}\,ed(v))$: если данные списки совпали, то соответствующие им узлы требуется объединить (слить вместе). Сложность алгоритма: $\text{O}(\log n + k)$.

После этого процесс удаления продолжается уже на узле $InSucc(v)$. Таким же образом процесс продолжается вплоть до попадания в узел w с диапазоном $[p^*,q]$, где p* – значение

большее или равное p. Чтобы завершить удаление, необходимо сравнить диапазон узла w с диапазоном его последователя $InSucc(v)$, и в случае совпадения узлы нужно слить вместе.
Каждый раз, когда узел сливается с его последователем или предшественником (в соответствии с последовательностью узлов при центрированном обходе), необходимо произвести вращения, как и в случае обычных AVL-деревьев.

Обработка секущих запросов

Дана точка одномерная точка p, требуется определить все содержащие ее отрезки, хранящиеся в дереве. Иными словами, нужно определить множество всех таких отрезков, которые пересекают прямую $x = p$. В случае стандартных деревьев отрезков ответ приходится собирать из нескольких узлов дерева; в случае описываемой структуры - только в одном узле. То есть просто нужно найти узел v, диапазон которого включает точку p, и ответом на запрос будет все множество $list(v)$. Делается это с помощью стандартного алгоритма бинарного поиска с асимптотической сложностью $O(\log n)$.

Обработка диапазонных запросов

Данный запрос подразумевает, что нужно найти все такие отрезок, которые пересекаются с некоторым заданным. С использованием описываемой структуры это делается эффективнее, чем в случае стандартных деревьев отрезков: сначала с помощью бинарного поиска по дереву находится узел u, диапазон которого включает в себя начало искомого; далее выполняется последовательное передвижение ($InSucc(u)$) по узлам до тех пор, пока не попадем в узел v, диапазон которого включает в себя конец искомого. Делается это за время O $O(\log n) + k$, где k - количество узлов, содержащих ответ на запрос.

3.2. Переход к двухмерным данным

В дальнейшем описанное в предыдущем разделе двухсторонне прошитое бинарное дерево поиска будем называть *ДПБДПЦО первого порядка* (так как оно служит для выполнения операций над одномерными данными).

Определение. ДПБДПЦО второго порядка для множества осепараллельных прямоугольников R состоит из двух ДПБДПЦО-деревьев первого порядка, по одному для каждого из измерений, обозначаемых как Tx и Ty соответственно, а также сбалансированного по высоте двоичного дерева поиска (AVL-дерева) Tc; для ДПБДПЦО второго порядка выполняются следующие свойства:

1. Каждый узел дерева Tc представлен как $v(m, color)$, где m соответствует идентификатору прямоугольника в множестве R, а $color$ изначально равняется 0 (поле $color$ используется для формирования окончательного ответа на запрос).
2. Деревья Tx и Ty связаны с горизонтальными и вертикальными сегментами прямоугольников из R соответственно.
3. Каждый узел в Tx и Ty представлен как v([a, b], L), где [a, b] - это отрезок, ассоциируемый с узлом v, а L - это список указателей, ссылающихся на узлы дерева Tc следующим образом: если прямоугольник $r \in R$ $[a, b]$ покрывает по x или y-координате отрезок [a, b] соответствующего измерения в Tx или Ty соответственно, то в L существует указатель, ссылающийся на узел, содержащий идентификатор r в Tc.

Оконный запрос

Выполнение оконного запроса по отношению к объектам-прямоугольникам, хранящимся в ДПБДПЦО второго порядка, разделяется на две последовательные фазы:

1. Сначала выполняется соответствующий полученному оконному запросу одномерный диапазонный запрос для первой координаты (то есть диапазонный запрос выполняется для координаты x окна поиска Q применительно к дереву Tx). Результатом этого запроса является получение множества указателей на узлы дерева Tc, претендующие на попадание в результирующее множество исходного оконного запроса; данные прямоугольные объекты действительно пересекаются с окном запроса Q x-координате. Переходя с помощью к каждого из полученных указателей к соответствующим узлам дерева Tc, мы изменяем значение поля $color$ этих узлов на 1 (красный).

2. Выполняется аналогичная описанной выше процедура окрашивания в черный цвет ($color = 2$) узлов дерева Tc, полученных в результате выполнения диапазонного запроса для y-координаты окна Q к дереву Ty. При этом в черный цвет окрашиваются только те узлы дерева Tc, которые в результате первой фазы были покрашены в красный. При этом, при окрашивании очередного узла в черный цвет, содержащийся в нем идентификатор отрезка тут же добавляется к ответу на исходный оконный запрос.

Очевидно, что наибольшее время, которое может быть затрачено на выполнение оконного запроса, оценивается как $O(\log n + k)$, где k – количество объектов, добавленных в ответ.

Заключение

В данной статье был рассмотрен ряд наиболее эффективных структур данных и применимых к ним алгоритмов обработки данных, используемых для пространственного индексирования и ответов на различные типы пространственных запросов. Также была описана такая структура данных, как двухсторонне прошитое бинарное дерево поиска с центрированным обходом, его использование в случае двухмерных пространственных данных, приведена оценка сложности основных запросов, включающих вставку, удаление данных и оконный запрос. Было показано, что эти оценки достаточно положительны в сравнении с соответствующими оценками для других рассмотренных структур данных, что позволяет перейти к программной реализации ДПБДПЦО и его тестированию на реальных данных.

Литература

1. N. Beckmann, H. Kriegel, R. Schneider, B. Seeger. The r*tree-an efficient and robust access method for points and rectangles. In Proceedings of the ACM SIGMOD 19th Intl. Conf. Management of Data. (SIGMOD’ 1990), апрель 1990, с. 322.

2. K. S. Easwarakumar, T. Hema. Bits-tree-an efficient data structure for segment storage and query processing. arXiv:1501.03435, 2015.

3. Antonin Guttman. R-trees: A dynamic index structure for spatial searching, vol.14. ACM, 1984.

4. H. Haverkort, F. Walderween. Four dimensional hilbert curved for r-trees. International Journal of Computer Mathematics, 13:221–229, 1983.

5. L. Arge, M. de Berg, H. Haverkort, K.Yi. The priority r-tree: A practically efficient and worst-case optimal r-tree. ACM Transactions on Algorithms (TALG), 4(1):9, 2008.

6. Raphael A.Finkel, Jon Louis Bentley. Quad trees – a data structure for retrieval on composite trees. Acta informatica, 4(1):1-9, 1974.

УДК 004.031.42

ВЫБОР РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ И ОЦЕНКА ЕЁ ЭФФЕКТИВНОСТИ ДЛЯ ПРИМЕНЕНИЯ В РЕСТОРАННЫХ ПРИЛОЖЕНИЯХ

В. С. Щербакова

Воронежский государственный университет

Введение

Рекомендательная система — это программный комплекс, который определяет интересы и предпочтения пользователя и формирует предложения контента в соответствии с ними. В основе системы лежит основная информация о пользователи и его пользовательская активность в веб-сервисе вокруг которого мы хотим построить рекомендательную систему.

В современном мире, где пользователи сталкиваются с огромным количеством информации, выбор становится сложным и ресурсоемким процессом. Рекомендательные системы играют ключевую роль в решении этой проблемы, предоставляя пользователям персонализированные предложения на основе их предпочтений и поведения. Они широко используются в различных отраслях, включая ресторанный бизнес, где рекомендации помогают клиентам выбрать блюда и рестораны на основе их предпочтений и ограничений.

Для реализации ресторанного приложения с рекомендательной системой, необходимо ознакомиться с существующими методами по реализации подобных сервисов, выбрать подходящий метод для реализации, а также обосновать выбор и сделать вывод о его эффективности.

1. Классификация рекомендательных систем

На данный момент существует множество различных рекомендательных систем, выделим основные четыре вида:

- коллаборативная (collaborative filtering CF)
- контент-ориентированная (content-based)
- основанная на знаниях (knowledge-based)
- гибридная (hybrid)

1.1 Коллаборативная (collaborative filtering CF)

Коллаборативный подход является естественной альтернативой классическому подходу user-based, и почти полностью его повторяет, за исключением одного момента — применяется он к транспонированной матрице предпочтений. Т.е. ищет близкие товары, а не пользователей. Пользовательская коллаборативная фильтрация (user-based CF) ищет для каждого клиента группу наиболее похожих на него (в терминах предыдущих покупок) клиентов и усредняет их предпочтения. Эти усредненные предпочтения и служат рекомендациями для пользователя. В случае же с товарной коллаборативной фильтрацией (item-based CF) ближайшие соседи ищутся на множестве товаров — столбцов матрицы предпочтений. И усреднение происходит именно по ним.

Действительно, если продукты содержательно похожи, то скорее всего они либо одновременно нравятся, либо одновременно не нравятся. Поэтому когда мы видим, что у двух товаров оценки сильно коррелируют, это может говорить о том, что это товары-аналоги.

Данная система имеет следующие недостатки:

1. Когда пользователей много (почти всегда), задача поиска ближайшего соседа становится плохо вычислимой. Например, для 1 млн пользователей нужно рассчитать и хранить ~ 500 млрд расстояний. Если расстояние кодировать 8 байтами, это получается 4TB для одной только матрицы расстояний. Если мы делаем item-based, то сложность вычислений снижается, а матрица расстояний имеет размерность уже не (1 млн на 1 млн) а, например, (100 на 100) по количеству товаров.

2. Оценка близости товаров гораздо более точная, чем оценка близости пользователей. Это прямое следствие того, что пользователей обычно намного больше, чем товаров и следовательно стандартная ошибка при расчете корреляции товаров там существенно меньше. В данном приложении достаточно информации о товарах, чтобы сделать вывод.

3. В user-based варианте описания пользователей, как правило, сильно разрежены (товаров много, оценок мало). С одной стороны это помогает оптимизировать расчет — мы перемножаем только те элементы, где есть пересечение. Но с другой стороны — сколько соседей не бери, список товаров, которые в итоге можно порекомендовать, получается очень небольшим.

4. Проблема холодного старта в рекомендательных системах — одна из важнейших проблем, связанных с рекомендательными системами, заключается в том, что исходное количество доступных рейтингов обычно невелико. Что делать, если новоиспеченный пользователь пока не оценил товар, который только появился? В таких случаях затруднительно применять традиционные модели коллаборативной фильтрации. Контентные методы и экспертные системы справляются с проблемой холодного старта увереннее коллаборативных моделей, но и они не всегда есть в распоряжении. Тем не менее, данную проблему можно решить относительно просто, предлагая пользователем при регистрации заполнять анкету о предпочтениях и добавлять по их характеристикам в соответствующие группы. Также можно подключать скрипт рекомендации после необходимо набранного количества зарегистрированных пользователей или количества совершенных покупок пользователей в приложении.

1.2 Основанная на контенте (content-based)

Этот тип лежит в основе многих рекомендательных систем. В отличие от коллаборативной фильтрации, этап знакомства с пользователем опускается. Товары и услуги рекомендуются на основе знаний о них: категория, производитель, конкретные функции и т.п. В общем, применяют любые данные, которые можно собрать.

По такому принципу работают системы интернет-магазинов, онлайн-кинотеатров и других сервисов.

Создатели платформ используют этот тип систем, чтобы не потерять новых пользователей, данных о которых еще нет. Отсюда же вытекают два недостатка: первое время системы действуют неточно и требуется больше времени на реализацию.

Персональные рекомендации предполагают максимальное использование информации о самом пользователе, в первую очередь о его предыдущих покупках. Одним из первых появился подход content-based filtering. В рамках данного подхода описание товара сопоставляется с интересами пользователя, полученными из его предыдущих оценок. Чем больше товар этим интересам соответствует, тем выше оценивается потенциальная заинтересованность пользователя. Очевидное требование здесь — у всех товаров в каталоге должно быть описание.

Исторически предметом content-based рекомендаций чаще были товары с неструктурированным описанием: фильмы, книги, статьи и др. Такими признаками могут быть,

например, текстовые описания, рецензии, состав актеров и прочее. Однако ничто не мешает использовать и обычные числовые или категориальные признаки.

Неструктурированные признаки описываются типичным для текста способом — векторами в пространстве слов. Каждый элемент такого вектора — признак, потенциально характеризующий интерес пользователя. Аналогично, продукт — вектор в том же пространстве. По мере взаимодействия пользователя с системой (скажем, он покупает товар), векторные описания приобретенных им товаров объединяются (суммируются и нормализуются) в единый вектор и, таким образом, формируется вектор его интересов. Далее достаточно найти товар, описание которого наиболее близко к вектору интересов, т.е. решить задачу поиска n ближайших соседей. Не все элементы одинаково значимы: например, союзные слова, очевидно, не несут никакой полезной нагрузки. Поэтому при определении числа совпадающих элементов в двух векторах все измерения нужно предварительно взвешивать по их значимости.

1.3 Основанные на знаниях (knowledge-based)

Этот тип работает на основе знаний о какой-то предметной области: о пользователях, товарах и других, которые могут помочь в ранжировании. Как и в случае с «content-based», оценки других пользователей системы не учитывают. Есть несколько разновидностей: case-based (основанный на конкретных примерах) , demographic-based (основанные на социальных данных), critique-based (основанный на критике) и т.д.

Но есть и минус — для разработки этой системы требуется много времени и ресурсов.

1.4 Гибридные (hybrid)

У всех описанных ранее типов есть определенные недостатки. Комбинирование нескольких алгоритмов в рамках одной платформы позволяет если не устранить их полностью, то хотя бы минимизировать.

Крупные сервисы и интернет-магазины используют гибридные варианты. Универсальной инструкции и рекомендаций по реализации такого инструмента нет. Есть несколько распространенных типов комбинирования:

1. Реализация по отдельности коллаборативных и контентных алгоритмов и объединение их предположений;
2. Включение некоторых контентных правил в коллаборативную методику;
3. Включение некоторых коллаборативных правил в контентную методику;
4. Построение общей модели, включающей в себя правила обеих методик.

Обычно эти варианты берут в качестве основы и дополняют по собственному желанию и по критериям сферы деятельности. Как и в случае с knowledge-based (основанный на знаниях), основной недостаток гибридных систем — сложность разработки .

2. Основание при выборе типа рекомендательной системы

Благодаря правильно подобранным рекомендациям можно существенно улучшить продажи. Исходя из выше сказанного, в качестве рекомендательной системы для реализации разрабатываемого проекта выбрана система коллаборативная фильтрация с использованием косинусного сходства по формуле 1. на анализе имеющихся сведений о сходства между пользователями (memory-based).

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \qquad (1)$$

Этот подход обладает следующими преимуществами:

1. Не требуется хранить большие данные. Достаточно только задать описание ключевых характеристик объектов и пользователя.
2. Сравнительная простота в реализации.

3. Реализация рекомендательной системы

Несмотря на то, что в приложении имеется возможность самостоятельной оценки позиций меню, многие пользователи не всегда используют данную функцию. Для выявления предпочтений каждого пользователя вектор предпочтений формируется исходя из частоты заказа пользователем того или иного блюда. Пользователям со сходными векторами предпочтений будут рекомендованы позиции меню, характерные для данной группы пользователей. Поэтому для оценки «близости» предпочтений пользователей наиболее часто используется косинусное сходство.

При добавлении новой оценки вектор интересов обновляется инкрементально (только по тем элементам, которые изменились). При пересчете имеет смысл давать новым оценка чуть больше веса, поскольку предпочтения могут меняться.

В (табл. 1) показан пример матрицы предпочтений, где строки соответствуют пользователям,а столбцы — объектам.

Можно увидеть (табл.2), что чаще покупают Товар 4, и расстояние покупок Пользователя 1 и Пользователя 2 выше 0,5, а это значит, что косинусная близость стремится к 1 и предпочтения Пользователя 1 и Пользователя 2 схожи, поэтому система рекомендаций может предложить товар, который он не покупал, но был куплен ранее схожим пользователем.

Таблица 1

Матрица предпочтений

	Товар 1	Товар 2	Товар 3	Товар 4	Товар 5	Вектор-расстояния
Пользователь 1	2	0	5	7	8	11,9163753
Пользователь 2	1	0	1	5	0	5,19615242
Пользователь 3	0	1	0	0	0	1
Пользователь 4	0	1	0	0	0	1

Таблица 2

Расчет расстояния $cos(\Theta)$

	$A \cdot B$	$\cos(\Theta)$
Пользователь 1 — Пользователь 2	42	0.67830222
Пользователь 3 — Пользователь 4	0	0
Пользователь 4 — Пользователь 5	1	1

Ядро рекомендательной системы представляет собой отдельный скрипт на языке программирования Python, который можно запускать по расписанию. Данный скрипт на основе истории заказов формирует векторы предпочтений для каждого пользователя и заносит их в отдельную табличку базы данных. Затем по критерию косинусной близости пользователи разбиваются на группы по предпочтениям (то есть группы, в которых косинусное расстояние между векторами предпочтений не превышают определенный эмпирически оцененный коридор значений). Для каждой подгруппы пользователей формируется вектор рекомендаций, состоящий из упорядоченных по частоте заказов позиций меню, которые и будут рекомендованы данной подгруппе.

4. Оценка эффективности рекомендательной системы

Для того чтобы понять, эффективна ли используемая рекомендательная система, необходим механизм оценки качества этой системы. Тестирование можно проводить как онлайн, так и офлайн. При проведении онлайн тестирования, рекомендательная система просто используется на пользователях и собираются отзывы о качестве рекомендаций у реальных пользователей системы. Такое тестирование, как правило, дает хорошие результаты, так как отзывы о работе системы исходят от реальных пользователей. Однако эксперименты с пользователями обычно являются дорогостоящими, поэтому чаще вместо этого отдают предпочтение офлайн метрикам.

Для оценки эффективности рекомендательной системы использовались такие офлайн метрики, как средняя абсолютная ошибка (MAE) и среднеквадратичная ошибка (RMSE). Они измеряют разницу между реальными и прогнозируемыми рейтингами.

Средняя абсолютная ошибка (MAE) — измеряет среднюю величину ошибок в наборе прогнозов без учета их направления. Это среднее по тестовой выборке абсолютное различие между прогнозом и фактическим наблюдением, где все индивидуальные различия имеют одинаковый вес.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right|$$

Среднеквадратическая ошибка (RMSE) — измеряет среднюю величину ошибки, но представляет собой квадратный корень из среднеквадратичной величины различий между прогнозом и фактическим наблюдением.

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Поскольку ошибки возводятся в квадрат до их усреднения, RMSE придает относительно большой вес большим ошибкам. Это означает, что RMSE показывает более точную величину ошибки, особенно в ситуациях, когда большие ошибки особенно нежелательны.

Точность p позволяет узнать, сколько из предоставленных пользователю рекомендаций оказались релевантными. Данный показатель рассчитывается как доля верно подобранных рекомендаций (True Positive), деленная на сумму себя и ошибочно подобранных рекомендаций (False Positive).

$$p = \frac{TP}{TP + FP}$$

В ходе работы по оценке эффективности рекомендательной системы на основе косинусного сходства были получены результаты метрики со значением RMSE = 0.968, а это

обеспечивается высокую точность прогнозирования и удовлетворительное качество рекомендаций для пользователей ресторанных приложений.

Заключение

Рекомендательные системы являются одним из наиболее используемых методов машинного обучения и находят применение во многих областях, от экономики до Интернета вещей. В этой статье мы представили оценку эффективности коллаборативной рекомендательной системы на основе косинусного сходства для ресторанных приложений. Результаты показали, что такая система обеспечивает хорошую точность прогнозирования и может успешно использоваться для предоставления персонализированных рекомендаций пользователям, упрощая процесс выбора ресторанов и блюд. Косинусное сходство оказалось эффективным методом для определения схожести пользователей и формирования персонализированных предложений.

Таким образом, рекомендательная система на основе косинусного сходства может стать ценным инструментом для ресторанных приложений, помогая увеличить уровень удовлетворенности клиентов и улучшить продажи. В дальнейшем исследовании можно рассмотреть возможность использования других методов коллаборативной фильтрации или гибридных подходов, а также учитывать дополнительные аспекты для еще более точных и актуальных рекомендаций.

Литература

1. *Гомзин, А.Г.* Системы рекомендаций: обзор современных подходов / А.Г. Гомзин, А.В. Коршунов // Труды Института системного программирования РАН. – 2012. – том 22. – С. 401–417.
2. *Ricci, F.* Recommender Systems. Handbook / F. Ricci, L. Rokach, B. Shapira, P. Kantor – NY :Springer, 2015 – 1008 p.
3. Анатомия рекомендательных систем 2. – Режим доступа: https://habr.com/ru/company/lanit/blog/420499. – (Дата обращения: 26.05.22)
4. Анатомия рекомендательных систем. – Режим доступа: https://habr.com/ru/company/lanit/blog/421401. – (Дата обращения: 04.01.23)
5. Знакомство с рекомендательными системами. – Режим доступа: https://habr.com/ru/company/piter/blog/350346. – (Дата обращения: 04.01.23)
6. Рекомендательные системы: как помочь пользователю найти то, что ему нужно? – Режим доступа: https://habr.com/ru/company/productstar/blog/523686. – (Дата обращения: 08.01.23)
7. MAE and RMSE – Which Metric is Better? – Режим доступа: https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d. – (Дата обращения: 18.04.23)
8. *Dietmar Jannach* Recommender Systems: An Introduction / D. Jannach, M. Zanker, A. Felfering, G. Friedrich, – M.: Cambridge University Press, 2011. – 352 с.

УДК 519.854.2

ПРИБЛИЖЁННОЕ РЕШЕНИЕ ЗАДАЧИ КОММИВОЯЖЁРА С ИСПОЛЬЗОВАНИЕМ ВЕНГЕРСКОГО МЕТОДА

С. А. Яковлева, О. А. Медведева

Воронежский государственный университет

Введение

Одной из важных задач транспортной логистики является задача коммивояжёра, целью которой является поиск оптимального маршрута движения разъездного посредника: он должен посетить все объекты задания за кратчайший срок и с наименьшими затратами.

Для решения задачи коммивояжёра существуют точные и эвристические методы. Точные алгоритмы производят полный перебор всех вариантов, к ним относят метод полного перебора и метод ветвей и границ. Задача коммивояжёра является NP-трудной, следовательно, точные методы имеют экспоненциальную стоимость по времени и при большом количестве объектов работают долго. Эвристические методы находят приближенное решение, но на больших данных работают гораздо быстрее точных, поэтому их разработка является актуальной задачей.

В данной работе будет предложен эвристический алгоритм для решения задачи коммивояжёра, основанный на применении венгерского метода решения задачи о назначениях.

1. Постановка и математическая модель задачи коммивояжёра

Пусть имеются n пунктов. При этом известна матрица затрат $\mathbf{C}$ размера $n \times n$, где c_{ij} — расстояния (или стоимости проезда, расход горючего и т.д.) от i-го пункта до j-го. Требуется построить замкнутый маршрут объезда всех пунктов, причем каждый пункт можно посетить только один раз. Связанные с этим суммарные затраты должны быть минимальными [2].

Для получения математической записи необходимо ввести n^2 переменных x_{ij}, $i = \overline{1,n}$, $j = \overline{1,n}$ следующим образом: $x_{ij} = 1$, если путь из пункта i в пункт j входит в маршрут, и $x_{ij} = 0$, в противном случае.

Математическая модель задачи в матричном представлении имеет следующий вид

$$L(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \to \min \tag{1}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \ \forall j = \overline{1,n}, \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \ \forall i = \overline{1,n}, \tag{3}$$

$$x_{ij} = \{0,1\}, \ i, j = \overline{1,n}. \tag{4}$$

С дополнительным условием отсутствия подциклов.

2. Эвристический алгоритм решения

Предлагается эвристический алгоритм решения задачи коммивояжёра. Его идея заключается в том, чтобы решать задачу коммивояжёра с матрицей затрат $\mathbf{C}$ как задачу о назначениях. Затем по полученной матрице назначений $\mathbf{X}$ построить маршрут. В общем случае получится несколько неполных замкнутых маршрутов (подциклов). В статье предлагается алгоритм перестроения матрицы $\mathbf{X}$ с целью получения полного замкнутого маршрута. Рассмотрим подробнее этапы построения результирующего маршрута.

Пусть задана матрица стоимостей $\mathbf{C}$. Так как в искомое решение задачи коммивояжёра не могут входить диагональные элементы, введем для них штраф $M > n \cdot \max c_{ij}$. Необходимо создать копию $\mathbf{C}'$ матрицы стоимостей $\mathbf{C}$, диагональным элементам которой присвоить штраф M. Решается задача о назначениях с запретами [1] с матрицей $\mathbf{C}'$ венгерским методом, ответ записывается в матрицу $\mathbf{X}$, $x_{ij} = 1$, если путь из пункта i в пункт j входит в маршрут, и $x_{ij} = 0$, в противном случае. Для построения маршрутов по матрице $\mathbf{X}$ предлагается следующий алгоритм.

Алгоритм 1. Построение замкнутых маршрутов по матрице назначений $\mathbf{X}$

1. Задать количество маршрутов $K = 0$. Текущий пункт $i = 1$. V — множество индексов непросмотренных строк, $V = \{1, ..., n\}$.
2. Пусть i — начало маршрута, $K = K + 1$. A_k — множество, элементами которого являются пары индексов k-го маршрута, присвоить $A_k = \varnothing$.
3. В i-й строке матрицы $\mathbf{X}$ найти единицу и соответствующий ей индекс столбца j. Задать $V = V \setminus \{i\}$, $A_k = A_k \bigcup \{(i, j)\}$. Перейти к пункту $j : i = j$.
4. Если маршрут не замкнулся, перейти к шагу 3.
5. Если $V \neq \varnothing$, начать выписывать новый маршрут, выбрав $i \in V$, и перейти к шагу 2.

Если маршрут один, то есть $K = 1$, то задача коммивояжёра решена. В противном случае необходимо соединить K маршрутов в один, то есть перестроить $\mathbf{X}$ таким образом, чтобы по ней можно было построить только один полный замкнутый маршрут. Введем переменную $R = \infty$, в которой будет храниться минимальное значение стоимости найденного пути. Выберем из каждого множества A_k, $k = \overline{1, K}$ одну пару индексов $(i_m{}^k, j_m{}^k)$, где $m = \overline{1, |A_k|}$. Создадим подматрицу $\mathbf{D}$ матрицы $\mathbf{C}'$, состоящей из элементов матрицы $\mathbf{C}'$, стоящих на пересечении $i_m{}^k$ строк и $j_m{}^k$ столбцов, $k = \overline{1, K}$. На главной диагонали матрицы $\mathbf{D}$ располагаются те элементы, для которых в матрице $\mathbf{X}$ стоят единицы, то есть раннее найденное решение. Нужно найти решение отличное от ранее найденного с минимальной суммарной стоимостью. Для этого нужно решить задачу о назначениях с запретами с матрицей $\mathbf{D}'$, копией матрицы $\mathbf{D}$, диагональным элементам которой присваивается штраф M, чтобы не получить уже раннее найденное решение. Ответ записать в матрицу $\mathbf{X}'$. Посчитать стоимость найденного решения

$L(X') = \sum_{i=1}^{K}\sum_{j=1}^{K} D_{ij}x'_{ij}$. Если $R > L(X')$, то $R = L(X')$, запоминаем выбранные индексы строк $I = \{i_m{}^k, k = \overline{1,K}\}$ и столбцов $J = \{j_m{}^k, k = \overline{1,K}\}$, а также матрицу $\mathbf{X}^* = \mathbf{X}'$. Таким образом перебираются все возможные варианты выбора пар индексов $(i_m{}^k, j_m{}^k)$ из множеств A_k, $k = \overline{1,K}$. Остается перестроить матрицу $\mathbf{X}$ в соответствии с лучшим решением, а именно $\forall i \in I$, $\forall j \in J$: $x_{ij} = x^*{}_{ij}$. Далее применить алгоритм 1 и посчитать стоимость найденного маршрута $L(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_{ij}$.

Таким образом получен следующий приближенный алгоритм решения задачи коммивояжёра.

Алгоритм 2. Решение задачи коммивояжёра

1) Задать матрицу $\mathbf{C}'$ с элементами
$$c'_{ij} = \begin{cases} c_{ij}, \text{ если } i \neq j, \\ M, \text{ в противном случае,} \end{cases}$$
где $i = \overline{1,n}$, $j = \overline{1,n}$, M — штраф.
2) Решить задачу о назначениях с запретами с матрицей $\mathbf{C}'$ венгерским методом. Ответ записать в матрицу назначений $\mathbf{X}$,
$$x_{ij} = \begin{cases} 1, \text{ если путь из пункта } i \text{ в пункт } j \text{ входит в результирующий маршрут,} \\ 0, \text{ в противном случае,} \end{cases}$$
где $i = \overline{1,n}$, $j = \overline{1,n}$.
3) По матрице назначений $\mathbf{X}$ построить все замкнутые маршруты по алгоритму 1.
4) Если маршрут единственный $(K = 1)$, перейти к шагу 9, в противном случае задать $R = \infty$, $I = \varnothing$, $J = \varnothing$, $\mathbf{X}^*{}_{K\times K}$.
5) Из каждого множества A_k, $k = \overline{1,K}$, выбрать по одному элементу с номером m: $(i_m{}^k, j_m{}^k)$, $m = \overline{1,|A_k|}$. Составить матрицу $\mathbf{D}$ размера $K \times K$, состоящей из элементов матрицы $\mathbf{C}'$, стоящих на пересечении строк с выбранными индексами $i_m{}^k$, $k = \overline{1,K}$ и столбцов с индексами $j_m{}^k$, $k = \overline{1,K}$. Задать матрицу $\mathbf{D}'$ с элементами
$$d'_{ij} = \begin{cases} d_{ij}, \text{ если } i \neq j, \\ M, \text{ в противном случае,} \end{cases}$$
где $i = \overline{1,n}$, $j = \overline{1,n}$, M — штраф.
6) Решить задачу о назначениях с запретами с матрицей $\mathbf{D}'$ венгерским методом. Ответ записать в матрицу назначений $\mathbf{X}'$,
$$x'_{ij} = \begin{cases} 1, \text{ если путь из пункта } i \text{ в пункт } j \text{ входит в результирующий маршрут,} \\ 0, \text{ в противном случае,} \end{cases}$$
где $i = \overline{1,n}$, $j = \overline{1,n}$.

Посчитать стоимость найденного решения $L(X') = \sum_{i=1}^{K}\sum_{j=1}^{K} D_{ij}x'_{ij}$. Если $R > L(X')$, то $R = L(X')$, $I = \{i_m{}^k, k = \overline{1,K}\}$, $J = \{j_m{}^k, k = \overline{1,K}\}$, $\mathbf{X}^* = \mathbf{X}'$.

7) Если рассмотрены все возможные варианты выбора $(i_k, j_k) \in A_k$, $k = \overline{1,K}$, перейти к шагу 9, в противном случае — к шагу 5.
8) $\forall i \in I\ \forall j \in J\ x_{ij} = x^*_{ij}$. Перейти к шагу 3.
9) Выписать ответ: замкнутый маршрут длины n соответствующий матрице назначений $\mathbf{X}'$ и его суммарная стоимость $L(X) = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}x_{ij}$.

3. Пример работы предложенного алгоритма

Пусть матрица стоимостей имеет вид:

$\mathbf{C} =$

	1	2	3	4	5	6
1	–	6	4	8	10	15
2	4	–	9	10	12	16
3	2	4	–	12	14	15
4	5	6	9	–	10	11
5	1	2	3	4	–	6
6	5	6	7	9	10	–

Шаг 1. Зададим $M = 96$:

$\mathbf{C}' =$

	1	2	3	4	5	6
1	96	6	4	8	10	15
2	4	96	9	10	12	16
3	2	4	96	12	14	15
4	5	6	9	96	10	11
5	1	2	3	4	96	6
6	5	6	7	9	10	96

Шаг 2. Решим задачу о назначениях с матрицей $\mathbf{C}'$. При этом матрица назначений $\mathbf{X}$ примет вид:

$\mathbf{X} =$

	1	2	3	4	5	6
1	0	0	1	0	0	0
2	1	0	0	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1
6	0	0	0	1	0	0

Шаг 3. Построим множества, задающие подциклы в найденном решении:

$A_1 = \{(1,3),(3,2),(2,1)\}$, $A_2 = \{(4,5),(5,6),(6,4)\}$.

Шаг 4. $R=\infty$, $I=\varnothing$, $J=\varnothing$, $\mathbf{X}^*_{K\times K}$

Шаг 5. Выберем элементы $(1,3)\in A_1$ и $(4,5)\in A_2$. Соответствующая матрица $\mathbf{D}'$ имеет вид:

$$\mathbf{D}' = \begin{array}{c|cc} & 3 & 5 \\ \hline 1 & 96 & 2 \\ 4 & 3 & 96 \end{array}$$

Шаг 6. Решим для матрицы $\mathbf{D}'$ задачу о назначениях. Матрица назначений $\mathbf{X}'$ имеет вид:

$$\mathbf{X}' = \begin{array}{c|cc} & 3 & 5 \\ \hline 1 & 0 & 1 \\ 4 & 1 & 0 \end{array}$$

Соответствующее значение целевой функции равно $L(X')=5$. Так как $R>L(X')$, то $R=5$, $I=\{1,4\}$, $J=\{3,5\}$, $\mathbf{X}^*=\mathbf{X}'$.

Шаг 5. Выберем элементы $(1,3)\in A_1$ и $(5,6)\in A_2$.

$$\mathbf{D}' = \begin{array}{c|cc} & 3 & 6 \\ \hline 1 & 96 & 6 \\ 5 & 2 & 96 \end{array}$$

Шаг 6.

$$\mathbf{X}' = \begin{array}{c|cc} & 3 & 6 \\ \hline 1 & 0 & 1 \\ 5 & 1 & 0 \end{array}$$

$L(X')=8$.

Повторяя предыдущие шаги, получим лучшее решение: $R=2$, $I=\{1,6\}$, $J=\{3,4\}$,

$$\mathbf{X}^* = \begin{array}{c|cc} & 3 & 4 \\ \hline 1 & 0 & 1 \\ 6 & 1 & 0 \end{array}.$$

Таким образом искомая матрица назначений примет вид:

$$\mathbf{X} = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 1 \\ 6 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Ей соответствует один замкнутый маршрут: $1-4-5-6-3-2-1$, стоимость маршрута $L(x)=39$.

Список литературы

1. Медведева О. А. Задача о назначениях, модели и алгоритмы: учебно-методическое пособие для вузов / С. Н. Медведев – Воронеж: Воронеж, 2018. – 40 с.

2. Чернышова Г. Д. Дискретная оптимизация: методическое пособие к курсу “Модели и методы дискретной оптимизации” / Каширина И. Л. – Воронеж: Воронеж, 2003. – 26

УДК 004.8

РАЗРАБОТКА ЭЛЕМЕНТОВ МЕДИЦИНСКОЙ СИСТЕМЫ ДЛЯ МОНИТОРИНГА СОСТОЯНИЯ ПАЦИЕНТОВ С ЛЕГОЧНЫМИ ЗАБОЛЕВАНИЯМИ

В. А. Яцков

Воронежский государственный университет

Введение

При диагностике и мониторинге состояния больных с легочными заболеваниями врачу необходима объективная информация о состоянии пациента. Наиболее часто для оценки состояния используются частота кашлей. В настоящее время врачи при оценке интенсивности и тяжести кашля основываются мнение пациентов, что является субъективной оценкой, так один и тот же кашель может быть описан двумя разными людьми по-разному. Информационная система, позволяющая проводить объективный мониторинг состояния пациента, могла бы помочь врачам корректнее оценивать состояние пациентов, вовремя корректировать лечение.

Известные устройства обладают рядом недостатков и не получили широкого распространения, поэтому исследование и разработка элементов таких устройств является актуальным.

Целью данной работы является исследование элементов медицинской информационной системы, которая позволяла бы регистрировать приступы кашля у людей, не обременяя их тяжелыми и неудобными устройствами, и, не ограничивая их местонахождение и движение во время мониторирования.

1. Классификация кашля

Подготовка данных

Кашель характеризуется рядом признаков: сокращение мышц грудной клетки, небольшие резкие наклоны вперед, резко выходящий изо рта воздух и характерный звук. Наибольший интерес при этом представляет анализ звуковых фрагментов. Это позволяет проводить мониторинг состояния пациентов без дополнительных датчиков, ограничивающих движение.

На аудиодорожках кашель предстает в виде резких всплесков громкости, как показано на рис. 1, однако эти всплески могут означать и просто другие громкие звуки: закрытие двери, падение предмета на пол или удар записывающего устройства о какую-либо поверхность. Для анализа таких фрагментов создадим и обучим нейронную сеть.

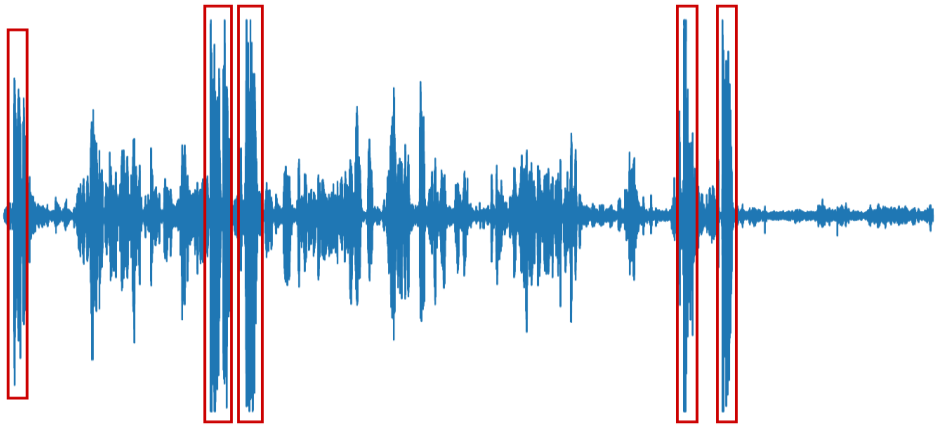


Рис. 1. Выделение фрагментов, предположительно, содержащих кашель из звуковой дорожки

Длительность практически любого отдельно взятого кашля не превышает пол секунды, поэтому из записи мы будет выделять фрагменты именно такой длины. Более того, звуки можно представить в различных вариациях: в виде амплитуд, как на рис.1, в виде спектрограммы или мел-кепстральных коэффициентов.

Спектрограмма получается в результате кратковременного преобразования Фурье и представляет собой набор интенсивностей частот на протяжении всего аудиофайла. Для наглядности можно изменить температурную карту (рис. 3). Мел-кепстральные коэффициенты (рис. 4) — это коэффициенты, которые описывают распределение мощности звука по частотам.

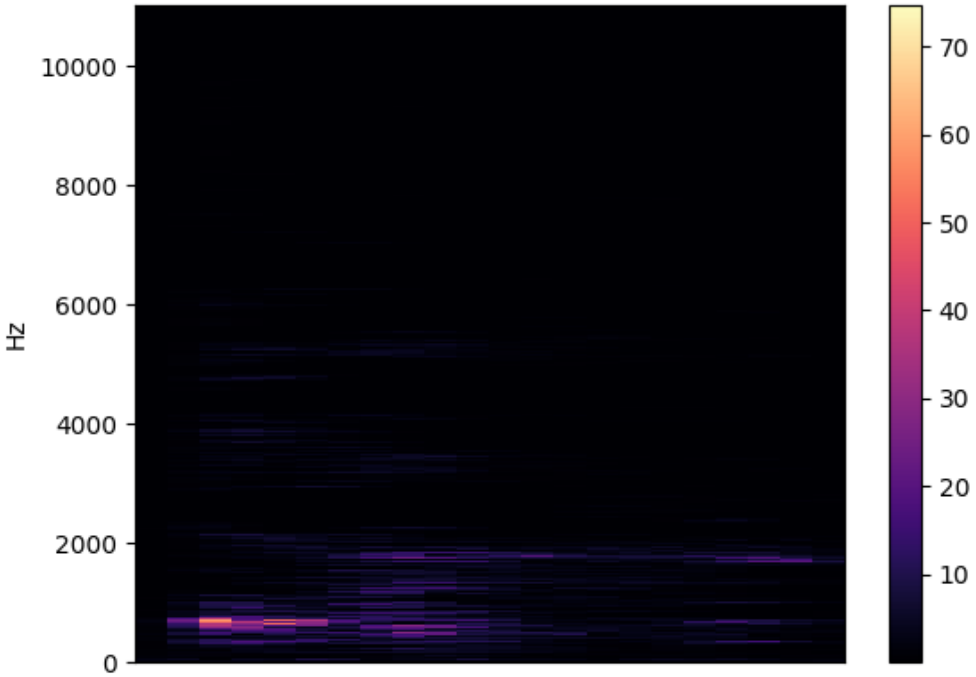


Рис. 2. Спектрограмма аудиофайла

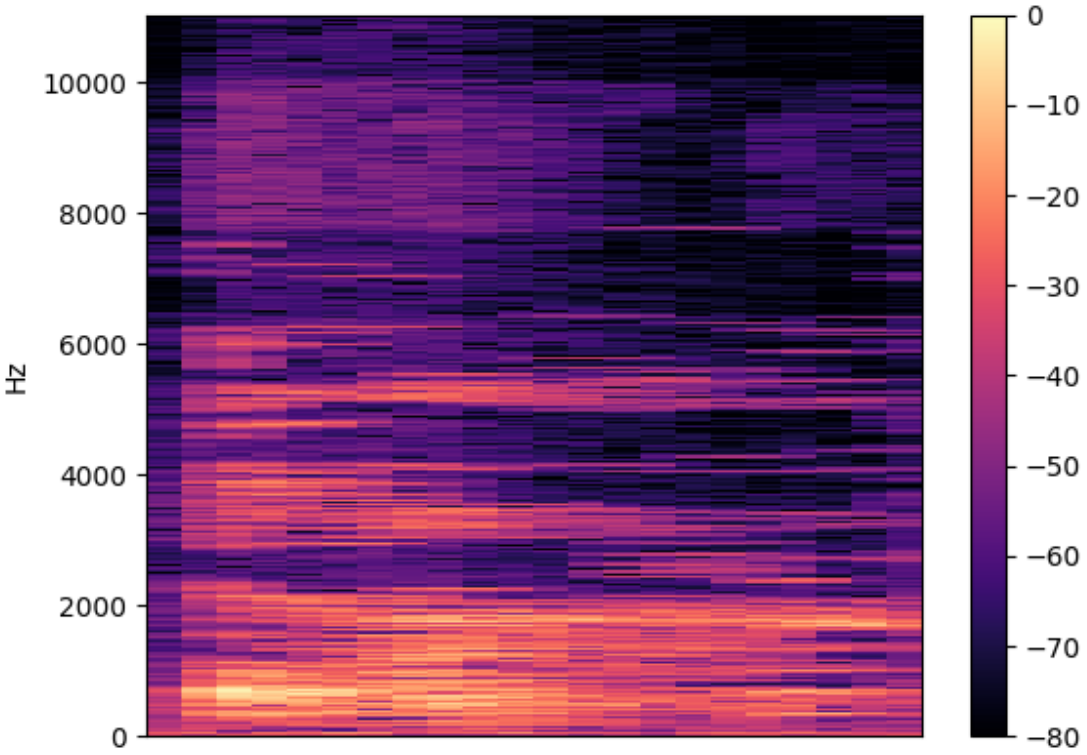


Рис. 3. Спектрограмма с измененной температурной картой

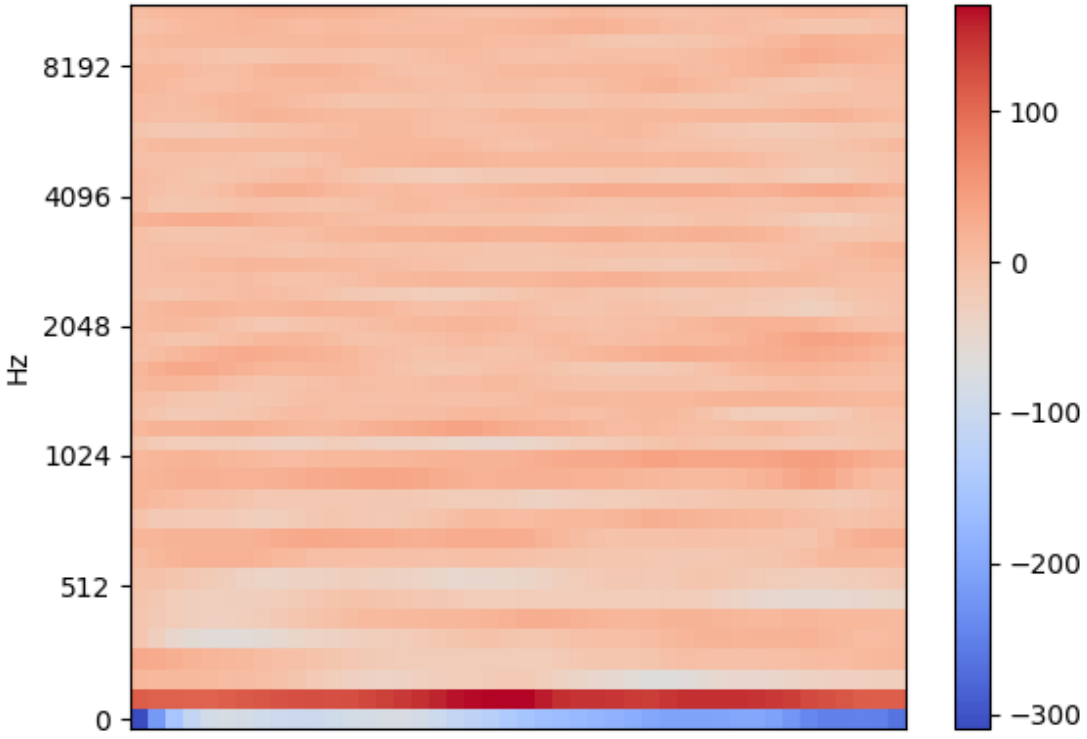


Рис. 4. Мел-кепстральные коэффициенты.

Для передачи данных в нейронную сеть можно использовать полученные изображения и решать задачу распознавания образов, построив сверточную нейронную сеть. Так как оные изображения — это визуализация данных о звуке, записанных в двумерный массив, то можно работать сразу с исходными массивами, а не с их визуальным представлением.

Выбор исходных данных

Из имеющихся вариантов представления данных необходимо выбрать тот, который позволит наилучшим образом отличать кашли от шумов. С этой целью обучаем нейронную сеть на каждом из наборов и смотрим в каком случае получается наилучший результат.

При этом также необходимо учитывать, что у врача часто отсутствует большой набор кашлей пациента. В работе рассмотрен подход, заключающийся в поэтапном включении ошибочно классифицированных шумов в обучающую выборку.

Выбор архитектуры нейронной сети

Для того чтобы определиться, с какой архитектурой нам продолжать работать, построим несколько моделей разной сложности и посмотрим на результаты их работы. Так как данные, с которыми мы имеем дело представлены в виде двумерных массивов, то обрабатывать их будем с помощью сверточной нейронной сети, вследствие чего основным образом мы можем изменять количество сверточных слоев, для получения нейронных сетей с различными архитектурами.

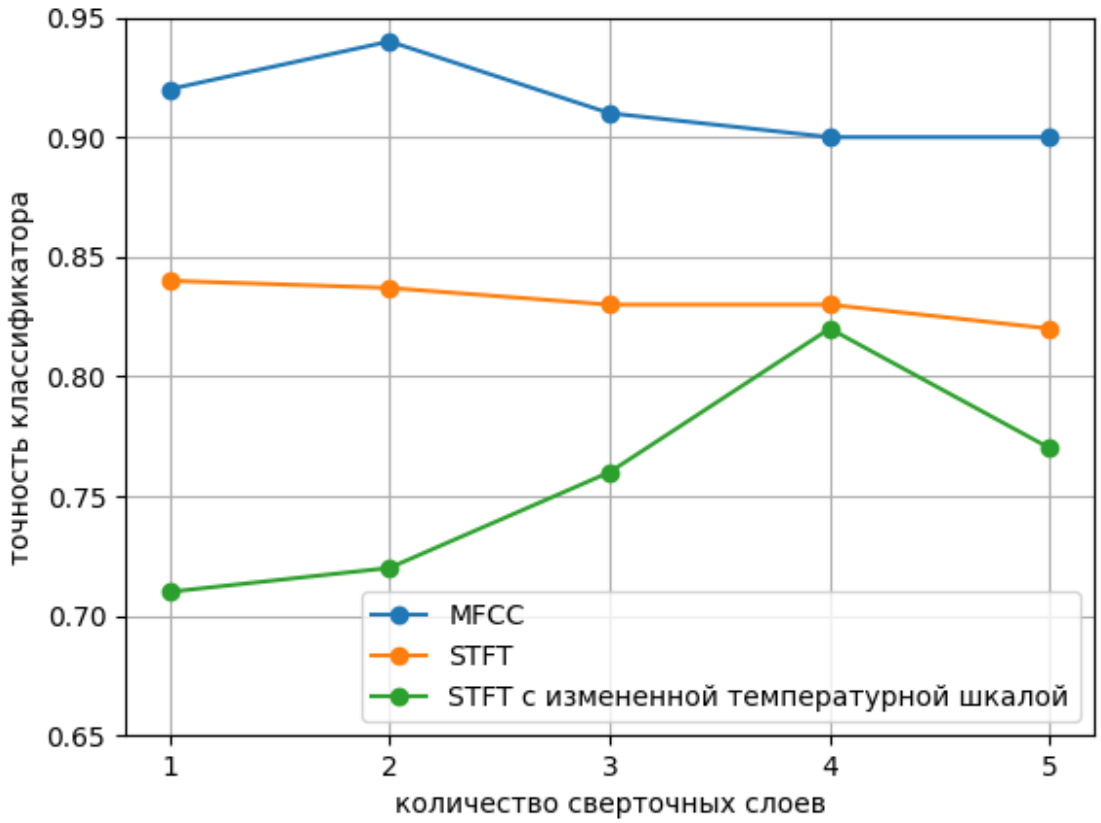


Рис. 5. Точность классификатора в различных условиях

Опираясь на ряд экспериментов, результаты которых изображены на рис. 5, можно сделать вывод о том, что больше всего для нашей задачи подходят массивы с мел-кепстральными коэффициентами и лучше всего они себя показывают при двухслойной

сверточной сети. Максимальная точность, которую удалось получить, составила 93.90%. При этом 8 посторонних шумов были помечены как кашель.

2. Избавление от ошибочной классификации шумов

Кроме того, что исследуемая нейронная сеть определяет не каждый кашель, она также помечает некоторые шумы, как кашель. Желательно уменьшить ошибочную классификацию шумов.

Для этого, проведено обучение нейронной сети в течении всего 4 эпох. Получено точность 83% и 25 шумов, определенных как кашель. Для улучшения свойств нейронной сети 5 случайных шумов из неверно определившихся добавляются в обучающую выборку. Обучение нейронной сети проводится еще 4 эпохи. Полученная точность 92% и 13 шумов, которые нейронная сеть классифицировала как кашель. Затем также дополнительно в обучающую выборку было добавлено 5 шумов. Как результат мы получили нейронную сеть, которая определяет отличает кашель от постороннего шума с точностью 97.83% и ошибочно записывает в число кашлей только 1 шум. График обучения модели, в сравнении с обучением без изменения обучающей выборки, представлен на рис. 6.

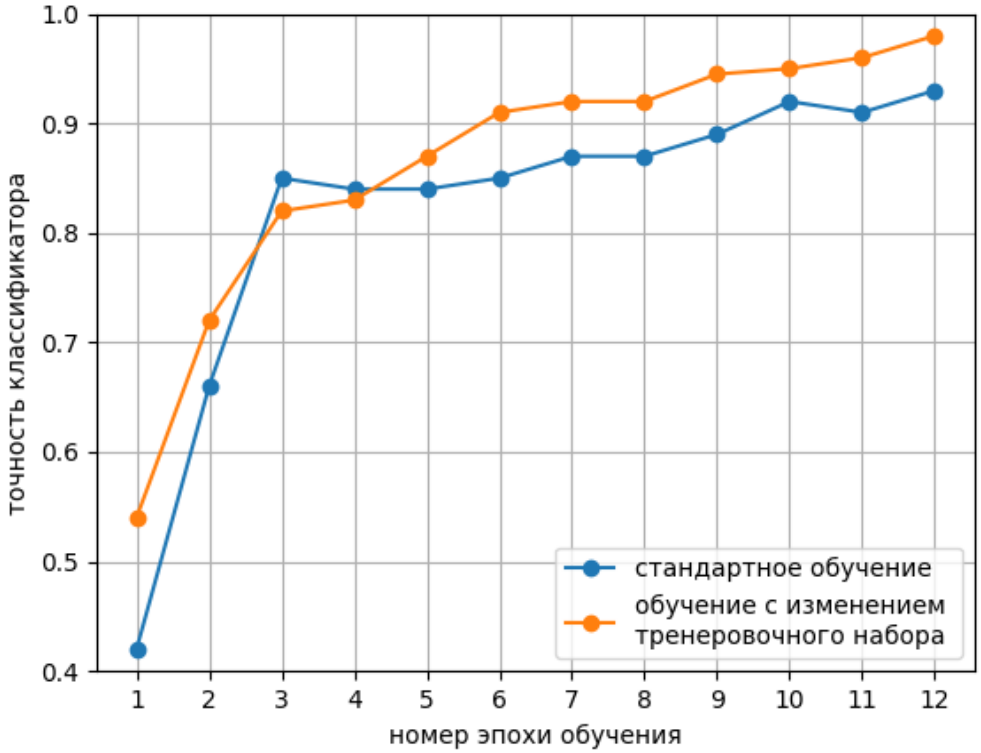


Рис. 6. График точности нейросети в процессе обучения

Заключение

В результате получилось создать классификатор на основе сверточной нейронной сети, которая позволяет с высокой точностью классифицировать кашли от посторонних шумов. Показана эффективность подхода с поэтапным добавлением ошибочно классифицированных

шумов в обучающую выборку. Использование которого можно в значительной степени повысить точность классификатора.

В дальнейшем планируется встроить данную нейронную сеть в мобильное или компьютерное приложение и выложить в открытый доступ.

Литература

1. Кинтцель, Т. Руководство программиста по работе со звуком / Кинтцель Т. –Москва : ДМК Пресс, 2000. – 432 с.
2. Аведьян Э.Д. Алгоритмы настройки многослойных нейронных сетей // Автоматика и телемеханика. – 1995. - № 4. - С. 106-118
3. Librosa—librosa 0.10.1dev documentation.— Режим доступа: https://librosa.org/doc/main/index.html
4. Nagesh Singh Chauhan: Audio Data Analysis Using Deep Learning with Python (Part 1). – Режим доступа: https://levelup.gitconnected.com/audio-data-analysis-using-deep-learning-part-1-7f6e08803f60. – (Дата обращения: 10.04.2023).

Содержание

Научное издание

Математика, информационные технологии, приложения

Сборник трудов
Межвузовской научной конференции
молодых ученых и студентов

Воронеж,
27 апреля 2023 г.

Подписано в печать 31.05.2022. Формат 60 × 84 / 8.
Усл. печ. л. 33,95. Тираж 100 экз. Заказ № ***.

Отпечатано с готового оригинал-макета

ООО Издательско-полиграфический центр
«Научная книга»
394030, г. Воронеж, ул. Никитинская, д. 38, оф. 308
Тел. +7 (473) 200-81-02, 200-81-04
http://www.n-kniga.ru; e-mail: zakaz@n-kniga.ru

Отпечатано в типографии ООО ИПЦ «Научная книга». 394026, г. Воронеж, Московский пр-т, 11б
Тел. +7 (473) 220-57-15, 238-02-38 http://www.n-kniga.ru; e-mail: typ@n-kniga.ru