МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

СБОРНИК СТУДЕНЧЕСКИХ НАУЧНЫХ РАБОТ ФАКУЛЬТЕТА КОМПЬЮТЕРНЫХ НАУК ВГУ

В ы п у с к 14

Часть 2
Научные работы студентов магистратуры

Под редакцией кандидата технических наук Д.Н. Борисова

Воронеж
Издательский дом ВГУ
2020

УДК 004.65 + 004.438.5
ББК Ч481(2)22
С23

Рекомендовано к печати
Ученым советом факультета компьютерных наук ВГУ

С23 **Сборник студенческих научных работ факультета компьютерных наук ВГУ.** Выпуск 14 : в 2 ч. / под ред. Д. Н. Борисова ; Воронежский государственный университет. – Воронеж : Издательский дом ВГУ, 2020.
ISBN 978-5-9273-3058-4

Часть 2 : Научные работы студентов магистратуры. – 324 с.
ISBN 978-5-9273-3060-7

В сборник включены научные работы студентов магистров факультета компьютерных наук ВГУ, выполненные в 2019–2020 гг. под руководством преподавателей факультета, представленные в виде докладов и рекомендованные к опубликованию оргкомитетами студенческих научных конференций.

УДК 004.65 + 004.438.5
ББК Ч481(2)22

ISBN 978-5-9273-3060-7 (ч. 2)
ISBN 978-5-9273-3058-4

Алгоритмы численного определения координат аналогов точки Ферма-Штейнера

Д. Г. Андреев

Студент магистрант

В. А. Родин

Профессор

Введение

Пусть M_k – точки в трехмерном пространстве с известными координатами $M_k = (x_k, y_k, z_k),\ k = 1,2,...,n$. Задача: «найти точку в пространстве (координаты этой точки), сумма расстояний от которой до заданных точек M_k будет минимальной». Эта задача Ферма–Штейнера насчитывает более 360 лет. Она получила большое прикладное значение после широкого применения сетей Штейнера [1].

Для числа точек – $n \geq 4$ аналитического решения определения координат точки Ф.–Ш. с оптимальной суммой расстояний до фиксированных *n* точек нет даже для плоскости.

В работе [1] получен алгоритм для численного решения задачи Ферма-Штейнера на плоскости. Для определения метрики рассматривались пространства l_p^N, $1 \leq p \leq \infty$, с нормой:

$$\left\|\vec{C}\right\|_{l_p} = \left(\sum_{k=1}^{N} |c_l|^p\right)^{1/p} \text{ для } 1 \leq p < \infty, \text{ где } \left\|\vec{C}\right\|_{l_\infty} = \max|c_k|.$$

Для $p = 2$ получаем метрику Эвклида, метрика l_1 в некоторых работах называется “метрикой Манхэттена”. Для решения различных прикладных задач в работе рассмотрены различные целевые функции с применением указанных норм для определения расстояний. Настоящая работа посвящена построению нового алгоритма, более простого для численного определения координат точки Ф.– Ш., но уже в трехмерном пространстве. Мы ограничимся только метрикой Эвклида [2].

Целевая функция, построенная для совокупности точек $M_k = (x_k, y_k, z_k),\ k = 1,2,...,n,$ имеет вид:

$$\Phi(M)=\Phi(x,y,z)=\sum_{k=1}^{n}\sqrt{(x-x_k)^2+(y-y_k)^2+(z-z_k)^2} \qquad (1)$$

1. Алгоритм численного решения задачи

Разобьем алгоритм на последовательные действия.

1. Действие первое – масштабирование исходных данных. Пусть $M_k=(x_k,y_k,z_k),\ k=1,2,...,n,$ – точки в трехмерном пространстве. Определим границы расположения этих точек. Пусть все точки лежат в первом квадранте и

$$\begin{aligned}&\min x_k=a,\quad \min y_k=b,\ \min z_k=c;\\ &\max(x_k-a)=A,\quad \max(y_k-b)=B,\quad \max(z_k-c)=C.\end{aligned} \qquad (2)$$

Тогда преобразование координат

$$\hat{x}=\frac{x-a}{A},\quad \hat{y}=\frac{y-b}{B},\quad \hat{z}=\frac{z-c}{C} \qquad (3)$$

переводит все точки в единичный куб – $[0,1]^3$. В этом кубе будет строиться направленный поиск. Обратное преобразование даст координаты решения задачи.

2. Действие второе – вычисление значений целевой функции в узлах сетки разбиения куба $[0,1]^3$. Каждую сторону куба разобьем на два равных интервала. При этом сам куб разобьется на восемь равных кубиков. Рассмотрим сеть, состоящую из центров этих восьми кубов меньшего размера:

$$m_{i,j,k}(1)=\left(\frac{1}{4}+\Delta\times i,\frac{1}{4}+\Delta\times j,\frac{1}{4}+\Delta\times k\right),\ \text{где}\ i=0,1;\ j=0,1;\ k=0,1.$$

В этих точках мы последовательно вычисляем значение целевой функции $\Phi\left(m_{i,j}(1)\right)$ и находим наименьшее значение этой функции. На этом первый шаг закончен.

Если точность вычисления достаточна, то эта точка и признается решением [3].

3. Действие третье – разбиение и вычисления теперь происходят в том из восьми кубов меньшего размера, в котором целевая функция достигла своего минимума.

По формулам (2) и (3) переходим к новому единичному кубу $[0,1]^3$ и повторяем действие второе. Узлами сетки разбиения нового куба будем называть центры новых кубиков еще меньшего размера –

$m_{i,j,k}(2)$. В этих точках мы последовательно вычисляем значение целевой функции $\Phi\left(m_{i,j}(2)\right)$ и находим наименьшее значение. Если точность достаточна, то эта точка и признается решением. Если нет, то все действия повторяются [4].

2. Блок-схема для компьютерной реализации решения задачи

Схема представляет собой итерационную схему, содержащую итерационный цикл, работающий до достижения нужной точности (рис. 1). Направленный алгоритм поиска гарантирует быструю достижимость нужной точности.

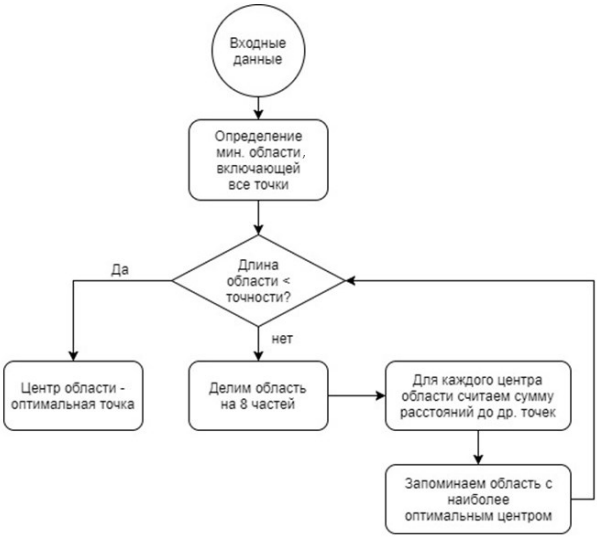


Рис. 1. Блок-схема решения задачи

3. Тестовые примеры визуализации

В этом пункте мы проверим операционную валидность алгоритма и полученной программы с помощью тестовых заданий. С позиции программирования – это задача визуализации действия программы. Исходные точки задавались двумя способами: 1) произвольно задавались координаты точек в пространстве (рис. 2), 2) случайным образом с помощью генерации случайных чисел (рис. 3) [5].

Данная программа была разработана на программной платформе .NET Framework. Ее основой является общеязыковая среда Common

Language Runtime (CLR), которая подходит для разных языков программирования. Для этой платформы был выбран объектно-ориентированный язык программирования C#.

Ниже приводим полученные результаты работы программы.

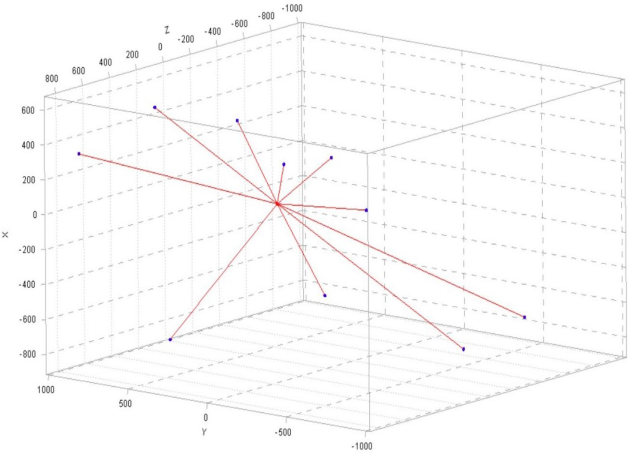


Рис. 2. Компьютерная визуализация работы программы при произвольно заданных точках

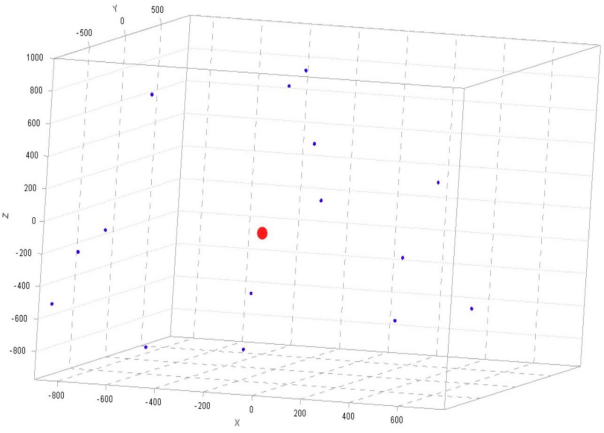


Рис. 3. Компьютерная визуализация работы программы путем генерации случайных чисел

4. Средства компьютерной реализации

Программа позволяет задать точки произвольно или сгенерировать их случайным образом. Также необходимо задать точность вычисления. В результате программа показывает все заданные точки и найденную оптимальную точку Ферма-Штейнера, затем вычисляет координаты этой точки и строит 3D-визуализацию.

Для визуализации используется компонент библиотеки ILNumerics (https://ilnumerics.net/). ILNumerics – это библиотека математических классов для разработчиков инфраструктуры общего языка (CLI) и доменного языка (DSL), который используется при реализации численных алгоритмов на платформе .NET. В то время как системы алгебры с графическими пользовательскими интерфейсами ориентированы на прототипирование алгоритмов, реализация таких алгоритмов в готовые к распространению приложения осуществляется с использованием сред разработки и языков программирования общего назначения (GPL).

Заключение

Определение координат точки Ферма–Штейнера для различных метрик имеют множество практических применений. Например, в южной части Вьетнама движение между небольшими населенными пунктами осуществляется по каналам и рекам. Эти населенные пункты сезонно затопляются и могут находиться под водой вплоть до 3-5 месяцев. Таким образом, задача стоит в определении оптимальных координат при строительстве поселений для работников рисовых плантаций.

Приведем другой пример. Если в фиксированных точках M_k разместить источники излучений, то в точке M интенсивность совместного эффекта будет максимальна. Задача определения такой точки актуальна для технической эстетики, дизайна и прикладных задач строительства, а также для планировки коммуникаций.

Список литературы

1. Иванов, А. О. Теория экстремальных сетей / А. О. Иванов, А. А. Тужилин. –М. ; Ижевск : Институт компьютерных исследований. – 2003. – 424 с.

2. Uteshev, A. Y. Stationary Points for the Family Fermat-Torricelli-Coulomb-like potential functions / A. Y. Uteshev, M. V. Yashina // Proc. 15th Workshop CASC, Berlin 2013. Springer. Lecture Notes in Computer Science. – 2013. –Vol. 8136. – P. 412-426.

3. Бондаренко, Е. С. Численное моделирование задач оптимального размещения обслуживающего объекта с использованием

аналогов точек Ферма-Штейнера / Е. С. Бондаренко, С. А. Гречаный, В. А. Родин // Вестник ВИ МВД России. – 2017. – № 2. – С. 154-161.

4. Родин, В. А. О точках Ферма–Штейнера в банаховых пространствах с различной нормой / В. А. Родин, Е. В. Родина // Системы управления и информационные технологии. – 2016. – №1(63). – С. 17-20.

5. Белый, Г. Ю. Построение оптимальной траектории для охраны акватории / Г. Ю. Белый, В. А. Родин // Вестник ВИ МВД. – 2015. – № 1. – С. 255-261.

Интеллектуальные системы поддержки принятия решений на основе анализа бизнес процессов

М. А. Андриянов

Студент магистрант

М. В. Корчагин

Доцент

Введение

Система поддержки принятия решений (СППР) – компьютерная автоматизированная система, целью которой является помощь людям, принимающим решение в сложных условиях для полного и объективного анализа предметной деятельности.

СППР возникли в результате слияния управленческих информационных систем и систем управления базами данных. Она решает две основные задачи. Во-первых, выбор наилучшего решения из множества возможных (оптимизация). Во-вторых, упорядочение возможных решений [1].

По взаимодействию с пользователем выделяют три вида СППР:

- пассивные – помогают в процессе принятия решений, но не могут выдвинуть конкретного предложения;
- активные – непосредственно участвуют в разработке правильного решения;
- кооперативные – предполагают взаимодействие СППР с пользователем [2].

На рисунке ниже приведена архитектурно-технологическая схема СППР.

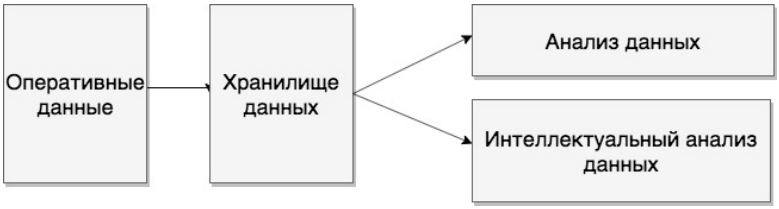


Рисунок. Структура СПР

Хранилище данных представляет собой банк данных определённой структуры, содержащий информацию о производственном процессе компании в историческом контексте. Главное назначение хранилища – обеспечивать быстрое выполнение произвольных аналитических запросов. Целью построения хранилища данных является интеграция, актуализация и согласование оперативных данных из разнородных источников для формирования единого непротиворечивого взгляда на объект управления в целом [3].

Анализ данных OLAP – сервис представляет собой инструмент для анализа больших объемов данных в режиме реального времени. Взаимодействуя с OLAP-системой, пользователь сможет осуществлять гибкий просмотр информации, получать произвольные срезы данных, и выполнять аналитические операции детализации. В основе концепции оперативной аналитической обработки (OLAP) лежит многомерное представление данных.

Оперативные данные OLTP – это способ организации БД, при котором система работает с небольшими по размеру транзакциями (оперативными хозяйственными операциями), идущими большим потоком. При этом пользователю требуется от системы максимально быстрое время ответа. Приложения OLTP, как правило, автоматизируют структурированные, повторяющиеся задачи обработки данных, такие как ввод заказов и банковские транзакции.

Интеллектуальный анализ данных Data Mining. При помощи средств добычи данных можно проводить глубокие исследования данных. Интеллектуальный анализ данных, ИАД (Data Mining) – это процесс поддержки принятия решений, основанный на поиске в данных скрытых закономерностей (шаблонов информации). При этом накопленные сведения автоматически обобщаются до информации, которая может быть охарактеризована как знания [4].

1. Методы использования разных СППР

В настоящее время рынок по разработке автоматизированных информационных систем принятия решений очень прогрессирует.

Для принятия решений СПР могут использовать разные методики.

В рамках изучения различных систем: социальных, экономических, природных, техногенных, для учета всех факторов из работы как единого объекта, важно учитывать совокупность внешних факторов. Но не так часто связи между частями можно учесть из-за нехватки всего объем данных, а для некоторых зада прогнозирования и имитации, данные вообще могут отсутствовать. Тогда нужные связи создаются путем оценок эксперта. Часто их реализуют в виде весовых параметров, применяемых для числовой оценки вклада текущего фактора в конечный результат.

Учет весовых параметров. Для данного учета используются разные подходы, в рамках которых уже реализовано много методик. Т.к. нет задачи обобщенного описания используемых вариантов выражения весовых коэффициентов, выполнил лишь анализ главных подходов.

Прямая расстановка. Эксперты уточняют веса факторов, базируясь на требованиях, к примеру, чтобы сумма весов была от 1% до 100%, хотя часто может иметь место другая постоянная, если это удобно для дальнейших расчетов. Часто данный процесс путают с передачей факторам некоторых значений по явной числовой шкале, но в таком случае лучше назвать такие факторы показателями значимости, а не весами, т.к. тогда имеют сравнительную оценку, а не их влияние совокупный итог. Так или иначе, подобным способом реализуют и весовые коэффициенты.

Трудности данного подхода заключены в возможности неявно держать в отдельных рамках все без исключения факторы, т.к., выделяя некое числовое значение любому фактору, эксперт должен при этом соотнести его с остальными. Сложность повышается в прогрессии с увеличением этих факторов.

Имеются и технические затруднения в работе специалиста, связанные с важностью периодического контроля текущей суммы весовых коэффициентов, чтобы не прийти к повышению указанной постоянной или передать крайним факторам оставшуюся большую часть. Если такое происходит, то принято пересчитать все отправленные коэффициенты, что можно сделать несколько раз, пока процесс обмена будет проходить. Число операций увеличивается с ростом числа факторов.

2. Ранжирование факторов

Такой подход позволяет облегчить экспертам работу, т.к. не нуждается в контроле общей суммы коэффициентов. В таком случае от экспертов необходимо ранжирование, т. е. упорядочивание

рассматриваемых факторов, формирующих объект, по степени выявления их свойств в порядке их минимизации или роста.

$$\left.\begin{matrix} R_{11}, R_{21}, & \text{K} & R_{n1} \\ \text{M} & & \text{M} \\ R_{1m}, R_{2m}, & \text{L} & R_{nm} \end{matrix}\right\}, \qquad (1)$$

где R_{ij} — ранг (место), данный фактору в ij -м экспертом в ряду из n - изученных объектов, поставленным этим экспертом по степени выражения анализируемого свойства. Возможно двум и более факторам иметь одинаковый ранг, но тогда он имеет вид дроби. Сводные оценки весовых коэффициентов получаются по итогу усреднения частных рангов по столбцам.

Плюсы подобного метода состоят в его простоте, но это не всегда тот случай, когда простота влияет положительно, поскольку усреднение рангов приводит к более грубым оценкам весовых коэффициентов в сравнении с остальными методиками. Также он не избавляет эксперта от обязанности контроля всех факторов, как и в случае прямой расстановки.

Передача коэффициентов факторам. В этом методе экспертам предлагается провести оценку факторов по некоторой балльной шкале, к примеру, от 1 до 10. В итоге получаем:

$$\left.\begin{matrix} y_{11}, y_{12}, \text{K}, y_{n1} \\ y_{12}, y_{22}, \text{K}, y_{n2} \\ \text{K K K K K K} \\ y_{1m}, y_{2m}, \text{K}, y_{nm} \end{matrix}\right\}, \qquad (2)$$

где y_{ij} – балльная оценка фактора, переданная от j -го эксперта, n – сумма факторов, m — количество экспертов.

Сводные оценки весовых коэффициентов находят зачатую методом подбора соответствующей регрессионной модели. Среднюю оценку ω_i весовых коэффициентов факторов получают по тривиальным формулам:

$$\omega_i = \frac{\sum_{j=1}^{m} \omega_{ij}}{\sum_{j=1}^{m} \sum_{i=1}^{n} \omega_{ij}} \qquad (3)$$

где ω_{ij} — вес i -го объекта, исходящий из оценок всех экспертов;

$$\omega_{ij} = \frac{\chi_{ij}}{\sum_{i=1}^{n} \chi_{ij}} \tag{4}$$

где χ_{ij} – оценка фактора i, приведенная экспертом j; n – число факторов, m – количество экспертов.

Такой метод в какой-то степени делает более слабой зависимость оценки отдельного фактора от остальных, но не полностью избавляет от нее, т.к. сопоставлять факторы необходимо, иначе не будет возможности расставить коэффициенты значимости корректно.

Заключение

Данная статья посвящена изучению систем поддержки принятия решений, их структуры и методов использования разных СППР. Были рассмотрены методы учета весовых параметров и ранжирования факторов. В процессе их рассмотрения, у каждого метода были выявлены свои достоинства и недостатки. При учете весовых параметров имеются технические затруднения, связанные с важностью периодического контроля текущей суммы весовых коэффициентов.

Список литературы

1. Системы поддержки принятия решений [Электронный ресурс]. – Режим доступа : http://bourabai.kz/tpoi/dss.htm

2. DSS – система поддержки принятия решений [Электронный ресурс]. – Режим доступа : http://pro-spo.ru/erp/1816-dss

3. Соловьев, Н. А. Основы теории принятия решений для программистов / Н .А. Соловьев, Е. Н. Чернопрудов, Д. А. Лесовой. – Оренбург : ОГУ, 2012. – 47 с.

4. Чубукова, И. А. Data Mining / И. А. Чубукова. – М. : Бином, 2006. – 195 с.

Адаптация информационной поддержки принятия решений в системе ПАО «Мосэнерго»

А. Н. Аргунова

Студент магистрант

В. А. Дурденко

Профессор

Введение

На сегодняшний день, в энергетической промышленности, предприятия самостоятельно разрабатывают свои стратегии развития. В том числе выполнение технического обслуживания и ремонта, которые регламентировались государственными нормативами, которые в свою очередь определяли планы (как, когда, кем и в каком объеме должны выполняться работы), то сегодня предприятие само определяет необходимость своевременного проведения мероприятий по техническому обслуживанию и ремонту (ТОиР).

Для сокращения затрат на ремонты и снижения потерь от простоев оборудования на многих предприятиях внедряют модуль SAP TOPO. Внедрение данного модуля позволяет руководителям технологических служб интегрироваться в корпоративную систему, получать и предоставлять более точную и своевременную информацию по всем задачам, связанным с планированием и учетом ТОиР. Решение ТОиР предназначено для автоматизации процессов планирования, управления и учета деятельности, связанной с техническим обслуживанием, ремонтом, послепродажным обслуживанием основных фондов.

Решение особенно востребовано на предприятиях со значительными производственными мощностями, повышенными требованиями к надежной и безотказной работе оборудования. Сотрудники таких предприятий ежедневно обрабатывают огромные объемы данных, что значительно усложняет планирование, подготовку и учет ремонтной деятельности. В рыночных условиях без автоматизации этих задач успешно вести бизнес не представляется возможным. Именно поэтому возникла необходимость в адаптации информационной поддержки принятия решений в системе ПАО «Мосэнерго».

1. Концептуальная модель системы информационной поддержки принятия решений

Концептуальная (содержательная) модель информационной системы поддержки принятия решений по планированию технического обслуживания и ремонта оборудования, представленная на рис. 1, разработана как абстрактная модель, отражающая общие характеристики моделируемой системы, ее структуру, свойства элементов и причинно-следственные связи, существенные для достижения цели моделирования.

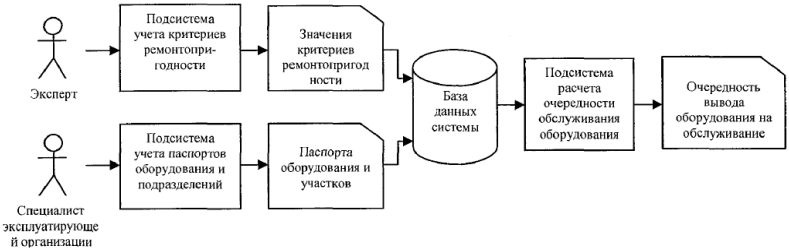


Рис. 1. Концептуальная модель работы автоматизированной системы поддержки принятия решений

Основными сущностями концептуальной модели являются подсистема учета критериев ремонтопригодности, подсистемы учета паспортов оборудования и подразделений и подсистемы расчета очередности обслуживания оборудования.

Концептуальная модель использовалась для дальнейшей разработки: расширенной функциональной структуры системы НИР, общей архитектуры системы и детальному описанию входных и выходных данных, как всей системы в целом, так и по подфункциям.

2. Функциональная структура системы информационной поддержки принятия решений

Был проведен анализ существующих методов принятия решений в условиях многокритериального ранжирования альтернатив [1]. Оптимальными методами для данной работы были выбраны следующие методы:

1. Метод Парето;
2. Лексикографический метод.

На основе данного анализа, составлена обобщенная функциональная структура системы информационной поддержки принятия решений. На рис. 2 наглядно отображена функциональная

структура системы, включающая главную функцию, подфункции второго уровня и подфункции третьего уровня.

Рис. 2. Функциональная структура системы поддержки принятия решений

Главной (базовой) функцией системы отражающей ее назначение и возможное применение является поддержка принятия решений для задачи технического обслуживания и ремонта в условиях динамического изменения потока входных данных.

СИППР включает в себя следующие подфункции второго уровня, наиболее полно характеризующие задачи системы:

1. Расчет очередности выполнения ремонтных работ, выполняется посредством использования неметрических многокритериальных методов принятия решений Парето и лексикографического [2]. В зависимости от моделируемой ситуации методы используются как вместе, так и по отдельности.

2. Формирование технико-экономической отчетности, выполняется по назначению пользователя СИППР в зависимости от вида выполняемых работ и необходимой для этого документации.

3. Работа с данными, выполняется при использовании встроенной БД, включающая в себя всю необходимую информацию об

оборудовании, подразделения и т.д. Данные вводятся пользователями системы, разбитые на две группы; эксперты и специалисты эксплуатирующей организации.

3. Структура базы данных системы информационной поддержки принятия решений

В основе моделирования базы данных системы информационной поддержки принятия решений для технического обслуживания и ремонта лежат представления пользователей системы: экспертов по проблемам технического обслуживания и ремонта, оценке качества работы и ремонтопригодности оборудования, специалистов эксплуатирующей организации, ответственных за планирование ремонтных работ на предприятии [3]. Данные вводятся в базу данных через программную оболочку системы на основании предоставления информации конечным пользователем. По уровню работы с данными различат два вида пользователей: эксперт и специалист эксплуатирующей организации.

Экспертом вводятся данные по оценкам оборудования согласно каждому критерию ремонтопригодности и приоритетам между самими критериями. Специалист эксплуатирующей организации вводит данные по оборудованию, подразделениям и нормативные данные.

При проектировании использовалась реляционная модель базы данных, т.к. она является простейшей и наиболее привычной формой представления данных в виде таблицы. К БД определены следующие требования, которым она должна удовлетворять:

1. обеспечивать возможность хранения и модификации больших объемов информации;
2. обеспечивать заданный уровень достоверности хранимой информации;
3. обеспечивать доступ к данным только пользователям с соответствующими полномочиями;
4. удовлетворять заданным требованиям по производительности при обработке запросов;
5. иметь возможность реорганизации и расширения при изменении границ ПО;
6. обеспечивать выдачу информации пользователю в различной форме;
7. обеспечивать простоту и удобство обращения внешних пользователей за информацией;
8. обеспечивать возможность одновременного обслуживания большого числа внешних пользователей.

Под базой данных обычно понимается именованная совокупность данных, отображающая состояние объектов и их отношений в рассматриваемой предметной области. Характерной чертой баз данных является постоянство: данные постоянно накапливаются и используются; состав и структура данных, необходимых для решения тех или иных прикладных задач, обычно постоянны и стабильны во времени; отдельные или даже все элементы данных могут меняться - но это и есть проявление постоянства - постоянная актуальность.

Самым распространенным способом представления структуры базы данных является ER - модель или модель сущность - связь. Представление данных в ER - модели осуществляется графически, в виде диаграммы сущностей и связей, состоящей из элементов трех основных типов: атрибутов, сущностей и связей [4].

На рис. 3 с помощью диаграммы сущность - связь отображена структура базы данных системы информационной поддержки принятия решений, спроектированная с помощью Microsoft SQL Server.

Рис. 3. Структура базы данных СИППР ТОиР

Заключение

Основным результатом данной работы является адаптация и разработка моделей, методов и алгоритмов информационной поддержки принятия решений в системе планирования технического обслуживания и ремонта оборудования на основе сформированного комплекса критериев экспертной оценки ремонтопригодности. В результате проведенных исследований создана автоматизированная система информационной поддержки принятия решений в модуле SAP TORO

для технического обслуживания и ремонта оборудования «АС ИППР ТОиР», позволяющая определять приоритеты оборудования при планировании работ ТОиР.

Список литературы

1. Ногин, В. Д. Принятие решений в многокритериальной среде: количественный подход / В. Д. Ногин. – М. : ФИЗМАТЛИТ, 2002. – 144 с.
2. Мельник, В. Ю. Применение неметрического метода Парето для задачи планирования технического обслуживания и ремонта / В. Ю. Мельник, В. А. Камаев, А. В. Кизим // Известия ВолгГТУ. – 2011. –№ 12. – С. 103-106.
3. Wireman, T. MRO inventory and purchasing / T. Wireman. – New York : Industrial Press, 2007. – 112 p.
4. Wireman, T. Successfully Utilizing CMMS/EAM Systems / T. Wireman. – New York : Industrial Press, 2008. – 238 p.

Разработка и моделирование систем массового обслуживания

И. А. Батюкова

Студент магистрант

В. А. Дурденко

Профессор

Введение

Во многих областях человеческой деятельности ставятся и решаются задачи, связанные с моделированием поведения некоторой системы в течение времени. Моделирование системы может преследовать разные цели: наблюдение за ходом процесса работы системы, оценка эффективности функционирования системы и ее оптимизация. Моделирование, обычно, применяется в тех случаях, когда проведение экспериментов над реальной системой невозможно или сильно затратно: например, создание прототипа для какого-либо эксперимента, либо из-за фактора времени, то есть расчеты, необходимые для развития проекта в будущем. Примерами таких систем являются телефонные системы, магазины, вычислительные комплексы и так далее.

При исследованиях в различных областях приходится встречать системы, которые предназначены для многократного использования при решении однотипных задач. Возникающие при этом процессы являются процессами обслуживания, а системы – системы массового обслуживания (СМО).

Наиболее важной проблемой разработки и моделирования СМО является недостаточность исследования предметной области для каждой конкретной системы, отсутствие возможности создания универсальной СМО для всех моделей одного типа. Данная работа посвящена изучению различных видов СМО и разработке алгоритма моделирования.

1. Система массового обслуживания

Системы массового обслуживания (СМО) – это математические модели таких реальных или проектируемых систем, функционирование которых можно рассматривать как последовательное взаимодействие неких дискретных объектов с элементами системы. Эти объекты, обычно поступающие в систему из внешних источников, некоторое время взаимодействуют с элементами системы по определенным правилам, а затем покидают систему [1].

В ходе формализации СМО традиционно представляются в виде схемы, представленной на рис. 1.

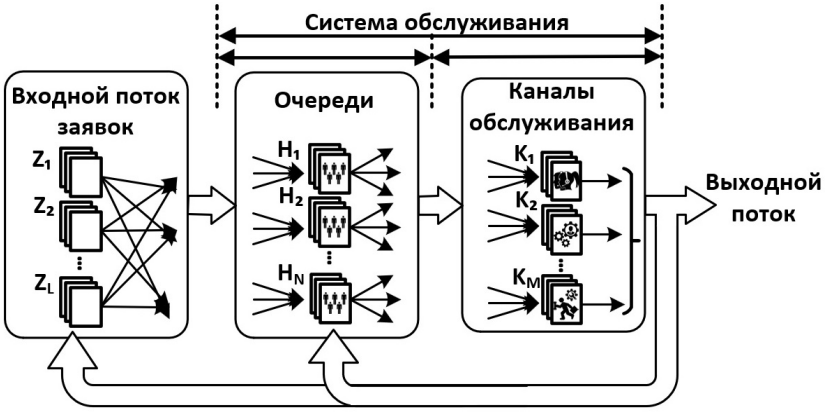


Рис. 1. Структурная схема СМО

Каждая СМО состоит из определенного числа обслуживающих единиц (приборов, устройств, пунктов, станций), которые называются

каналы обслуживания. Каналами могут быть линии связи, рабочие точки, вычислительные машины, продавцы и другие.

Все СМО делятся на два класса:

– разомкнутые СМО (входной поток заявок не зависит от процесса обслуживания, то есть нет обратной связи);

– замкнутые СМО (входной поток заявок зависит от числа заявок, находящихся на обслуживании, а заявки с выхода могут опять поступать на вход системы, то есть существует обратная связь).

2. СМО с отказами и с ожиданием

Заявки поступают в СМО случайным образом, то есть не регулярно, образуя так называемый случайный поток заявок. Обслуживание заявок также продолжается какое-то случайное время.

Случайный характер данного явления приводит к тому, что СМО оказывается загруженной неравномерно: в какие-то периоды времени скапливается достаточно большой поток заявок (они либо становятся в очередь, либо покидают СМО, получив отказ в обслуживании), в другие же периоды СМО работает с неполной загрузкой.

В связи с этим СМО делят на два основных типа: СМО с отказами и СМО с ожиданием (очередью).

В *СМО с отказами* (представлена на рис. 2) заявка, поступившая в момент, когда все каналы заняты, получает отказ, покидает СМО и в дальнейшем процессе обслуживания не участвует (например, заявка на телефонный разговор в момент, когда все каналы заняты, получает отказ и покидает СМО необслуженной). В СМО с ожиданием заявка, пришедшая в момент, когда все каналы заняты, не уходит, а становится в очередь на обслуживание.

Для СМО с отказами используют следующие показатели эффективности:

– абсолютная пропускная способность – среднее количество заявок, обслуживаемых за определенное время;

– относительная пропускная способность – отношение среднего количества обслуженных заявок к среднему числу поступивших заявок за определенное время;

– среднее число одновременно занятых каналов обслуживания и коэффициент их использования.

СМО с ожиданием подразделяются на разные виды в зависимости от того, как организована очередь: с ограниченной (представлена на рис. 3) или неограниченной длиной очереди (представлена на рис. 4), с ограниченным временем ожидания и так далее.

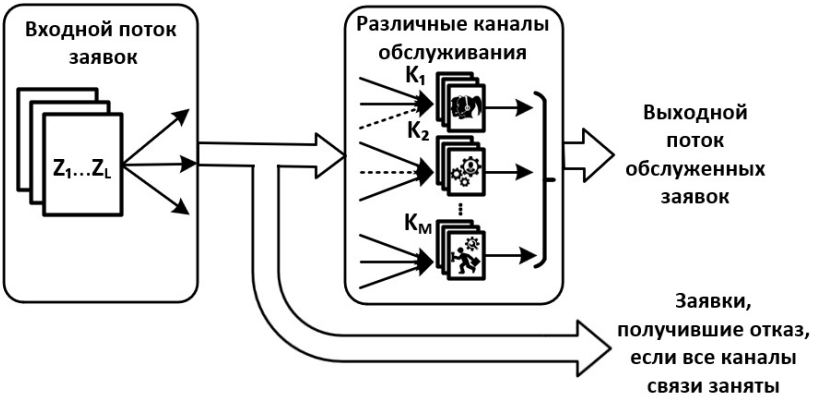


Рис. 2. СМО с отказами

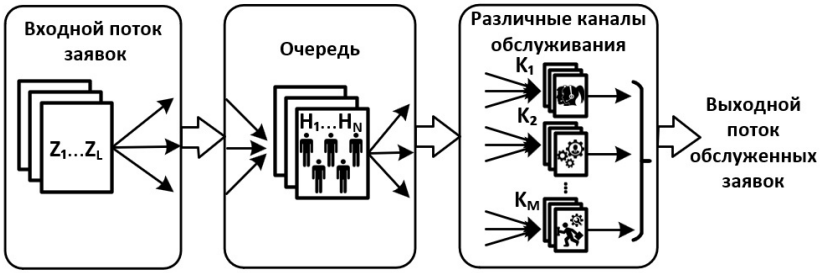


Рис. 3. СМО с ожиданием для неограниченной очереди

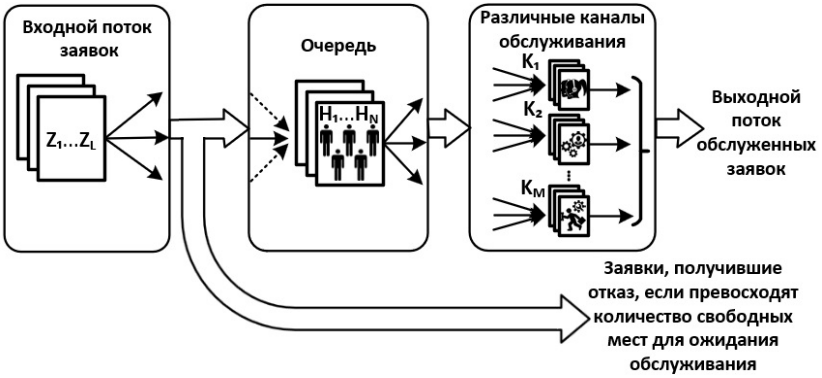


Рис. 4. СМО с ожиданием для ограниченной очереди

Очередь образуется в результате перегрузки приборов. В результате неограниченной очереди не происходит потери заявок. В конечной очереди устанавливается ограничение на количество свободных мест для ожидания обслуживания. Ограничения на длину очереди задаются условиями, при которых накопление и хранение заявок, ожидающих обслуживания, становится по некоторым соображением нецелесообразным или невозможным [2].

Для СМО с ожиданием используют следующие показатели эффективности:

– вероятность не превышения времени ожидания заявки в очереди заданного предельного значения;

– среднее количество заявок в очереди и в системе в целом (в очереди и на обслуживании);

– среднее время пребывания заявки в очереди и в системе в целом (в очереди и на обслуживании);

– среднее количество одновременно занятых каналов и коэффициент их использования.

Основной целью моделирования СМО является оценка эффективности в зависимости от структуры и параметров системы. Для этого необходимо правильно задавать и использовать вышеуказанные показатели эффективности, которые отличны для различных типов СМО.

3. Имитационное моделирование

Имитационное моделирование – один из самых распространенных методов исследования операций и теории управления.

Имитационная модель – это компьютерная программа, которая описывает структуру и воспроизводит поведение реальной системы во времени. Имитационная модель позволяет получать подробную статистику о различных аспектах функционирования системы в зависимости от входных данных [3].

Реализация технологий имитационного моделирования применительно к задачам исследования СМО предполагает:

1. построение алгоритмов и программных модулей, вырабатывающих реализации случайных потоков однородных и неоднородных событий – «генераторов» источников;

2. построение моделирующих алгоритмов и программных модулей, описывающих функционирование отдельных элементов, а также СМО в целом в соответствии с ее структурой и внутренними параметрами;

3. многократное воспроизведение входных потоков и общего процесса обслуживания, а также обработку получаемых данных в интересах оценки показателей эффективности данного типа СМО.

При моделировании работы СМО с приоритетной дисциплиной обслуживания, возможна ситуация, при которой низкоприоритетные заявки могут вообще не поступать на обслуживающий прибор, либо процент обслуженных низкоприоритетных заявок будет мал, такая ситуация возможна, когда интенсивность прихода заявок абсолютного и относительного приоритета высока и прибор занят обслуживанием заявок более высокого приоритета. Решением этой проблемы является использование динамических приоритетов.

Заключение

Данная работа посвящена изучению различных видов и алгоритма моделирования СМО. В настоящее время имитационное моделирование применяется в самых разных областях человеческой деятельности: в промышленности, на транспорте, в экономике, экологии, в сферах информационной безопасности и услуг, а также в сферах общественных, государственных и военных отношений.

Применение имитационного моделирования позволяет существенно снизить временные и материальные затраты. Использование имитационного моделирования с помощью электронно-вычислительных машин позволяет увеличить скорость получения оптимальных параметров системы, проведения необходимых экспериментов и модификаций.

Список литературы

1. Ослин Б. Г. Моделирование. Имитационное моделирование СМО : учебное пособие / Б. Г. Ослин. – Томск : Изд-во Томского политехнического университета, 2010. – 128 с.

2. Климов Г. П. Теория массового обслуживания / Г. П. Климов. – 2-е издание, переработанное. – М. : Издательство Московского университета, 2011. – 312 с.

3. Имитационное моделирование. [Электронный ресурс] : Официальный сайт платформы для имитационного моделирования AnyLogic. – Режим доступа : https://www.anylogic.ru/use-of-simulation/

Построение непрерывного скелета многоугольной фигуры

А. С. Богданова

Студент магистрант

В. В. Фертиков

Доцент

Введение

В настоящее время цифровая обработка, анализ и распознавание изображений являются одними из наиболее интенсивно развивающихся направлений научных исследований, что обуславливается широким распространением и практическим использованием систем машинного зрения, видеонаблюдения, моделирования графических объектов в различных отраслях науки и техники.

Одной из важных проблем при обработке и анализе изображений является затрата времени на выполнение алгоритмов. В данной статье предлагается эффективный по вычислительной сложности решение задачи построения непрерывного скелета многоугольной фигуры, граница которой состоит из конечного числа простых непересекающихся многоугольников.

1. Подготовка построения контура

Рассмотрим для анализа бинарное изображение рис. 1, состоящее из произвольного множества растеризованных объектов. Такие объекты могут иметь «многоугольные дыры» и тем самым образовывать вложенность отверстий.

Понятие непрерывного скелетного описания в математическом виде определена для замкнутой плоской области: это множество тех ее точек, для которых существует не менее двух равноудаленных ближайших точек границы области.

В данной работе для представления непрерывной сцены используется подход [1] с ассиметричной треугольной структурой соседства. Это обусловлен тем, что такая структура имеет перекрестки, где существует единственная диагональная пара точек, соединенная ребром, являющаяся смежной. Из этого следует, что треугольная решетка дает возможность построить эквивалентную непрерывную сцены для любой дискретной сцены.

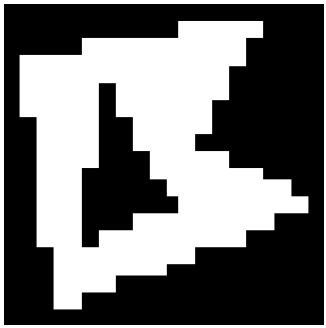


Рис. 1. Пример бинарного изображения

На основе триангулярного графа соседства пикселей образуются варианты топологической смежности дискретных фигур: гексагональной, объектной и компонентной.

В данной работе реализованы все три вида прослеживания границ на базе этих структур. На рис. 2 представлена прослеживание границ подвижным мостом при компонентной смежности.

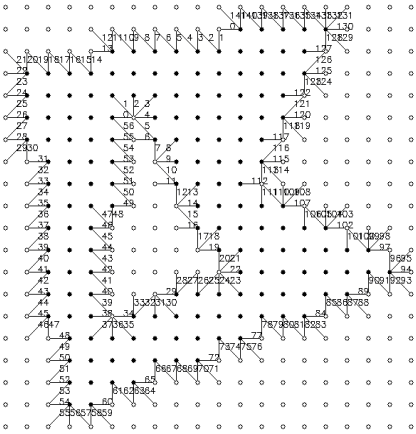


Рис. 2. Пример маршрута подвижного моста при компонентной смежности

Для дальнейшего представления объекта для каждого контура производится построение многоугольника минимального периметра [1] на рис. 3, а затем производится компоновка контуров в рамках единой структуры.

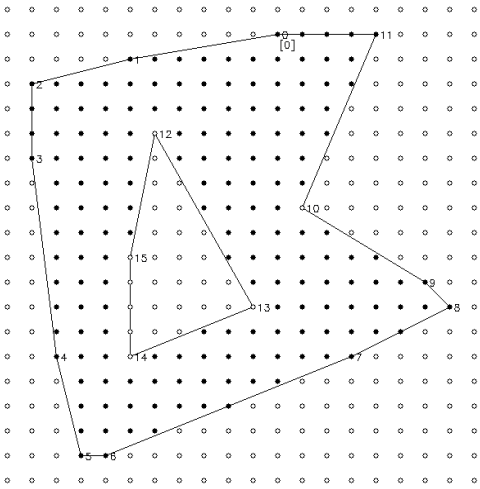


Рис. 3. Пример разделяющего многоугольника минимального периметра

2. Метод построения скелета

Полученная структура контурного представления используется для построения непрерывного скелета. Для построения скелета используется обобщенная диаграмма Вороного многоугольной фигуры. Диаграмма Вороного описывается плоским графом, ребрами которого являются бисекторы пар сайтов, а вершинами – точки пересечения этих бисекторов. В свою очередь бисектор является линией, разделяющей смежные ячейки Вороного двух сайтов. В зависимости от типа сайтов бисектор либо линейный, либо параболический. Для построения диаграммы необходимо вычислить бисекторы и точки их пересечения.

Для реализации построения диаграммы Вороного был выбран жадный алгоритм [1], так как он прост в исполнении. Такой метод имеет квадратичную вычислительную сложность относительно исходного числа сайтов.

Суть алгоритма заключается в том, чтобы последовательно находить все вершины и рёбра диаграммы Вороного. Так как такая диаграмма является связным графом, то этот процесс реализован путем последовательного присоединения к уже построенной части диаграммы новых ее элементов – ребер и вершин.

На рис. 4 представлен непрерывный скелет многоугольной фигуры.

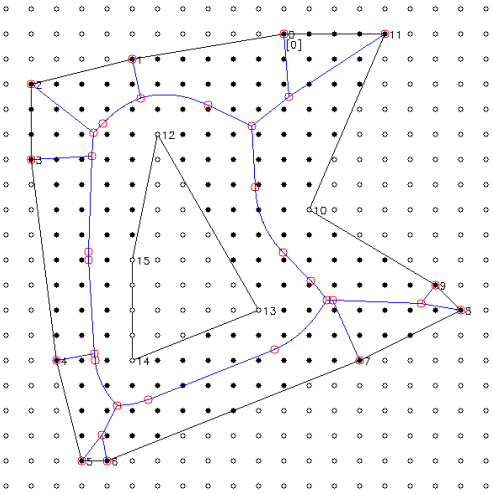


Рис. 4. Пример непрерывного скелета многоугольной фигуры

В данной работе жадный алгоритм [1] будет использоваться как основной инструмент. С точки зрения практики этот алгоритм не всегда приемлемо использовать, так как он ограничен небольшим количеством сайтов. Чтобы избежать ограничений предлагается предварительно проанализировать смежность сайтов, которые значительно уменьшат множество пробных сайтов для построения пустых кругов.

Для предварительного анализа смежности сайтов предлагается граф Делоне. Графом смежности (граф Делоне) многоугольной фигуры называется граф, у которого множество вершин состоит из сайтов фигуры, а подмножество ребер содержит все пары смежных сайтов.

3. Метод дискретного преобразования расстояний

Для реализации построения обобщенного графа Делоне предлагается подход из статьи [2] с использованием дискретной дистанционной картой. Такие алгоритмы относятся к категории дискретного преобразования расстояний.

Предварительно производиться растеризация сайтов с присвоением им уникальных номеров. Далее на основе алгоритма преобразования расстояний вычисляется дистанционная карта, каждый элемент который помечен номером ближайшего к нему сайта. Таким обазом получен дискретный аналог диаграммы Вороного просматривается с целью поиска границ сайтов.

Заключение

Данная статья посвящена разработке и реализации алгоритма построение непрерывного скелета. Рассмотрены алгоритмы прослеживания границ контуров, построение непрерывного скелета на основе диаграммы вороного и графа Делоне. Использование данного решения позволит увеличить скорость выполнения алгоритма.

Список литературы

1. Местецкий, Л. С. Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляторы / Л. С. Местецкий. – М. : Физматлит, 2009. – 286 с.

2. Фертиков, В. В. Непрерывное скелетное описание с преминением дискретного преобразования расстояний / В.В.Фертиков // Информатика: проблемы, методология, технологии : сб. м. XVII международной научно-методической конф. – Воронеж, 2017. – С. 179-183.

Исследование эффективности методов машинного обучения в задачах прогнозирования

К. В. Бородкин

Студент магистрант

Р. Р. Отырба

Студент магистрант

А. С. Пешков

Студент магистрант

М. А. Дрюченко

Доцент

Введение

В настоящее время, с распространением машинного обучения, появляется все больше решений для задач прогнозирования. Начинались они, с классических, линейных методов, таких как логистическая регрессия, наивный байесовский классификатор, k-ближайших соседей и другие. Далее появились ансамблевые методы, которые сочетают в

себе, как правило, деревья принятия решений, а с недавнего времени, большую популярность обрели нейронные сети. В таком многообразии становится тяжело разбираться в выборе нужного алгоритма для конкретной задачи. Данная статья представляет собой попытку разобраться, почему некоторые алгоритмы показывают себя хуже или лучше с разных точек зрения. В частности, будет рассмотрен метод опорных векторов, градиентный бустинг на основе решающих деревьях и классическая глубокая полносвязная нейронная сеть.

Данное исследование не претендует на утверждение, какой метод однозначно лучше или хуже другого, оно лишь предложит несколько общих рекомендаций, так как невозможно для любой задачи прогнозирования прописать эвристики.

В процессе этой работы будут рассмотрены алгоритмы с точки зрения требований к вычислительной мощности, зависимости от размерности набора данных, и, естественно, точности прогнозирования по метрике AUC-ROC. Над всеми методами будут проводиться эксперименты на задачах прогнозирования на пяти наборах реальных статистических данных, взятых с ресурса соревнований. Все пять наборов различаются как по составу, так и по объему.

1. Условия экспериментов

Для репрезентативности экспериментов, следует уточнить его условия. Как уже было упомянуто ранее, в процессе эксперимента будут рассматриваться следующие представители машинного обучения:

– Light Gradient Boosting Machine – имплементация градиентного бустинга над решающими деревьями.

– Support Vector Machine – классический классификатор, идея которого заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом.

– Deep Dense Neural Network – глубокая нейронная сеть с полнозвязными соединениями нейронов в слоях.

Любой эксперимент машинного обучения нуждается во множестве наборов данных для обучения. В нашем случае были отобраны пять наборов для задач прогнозирования с ресурса соревнований Kaggle. Ниже описаны их предназначение и некоторые характеристики (табл. 1).

1. Первый набор – содержит обезличенную информацию для прогноза «Кто совершит следующую финансовую транзакцию?».

2. Второй набор – включает в себя обезличенную информацию для прогноза «Какой покупатель остался доволен?».

3. Третий набор – предназначен для решения задачи предсказания потребности сотрудника в доступе к сетевым ресурсам.

4. Четвертый набор – содержит информацию для прогноза «Какие клиенты приобретут указанный страховой план?»

5. Пятый набор – предоставляет тривиальную задачу прогнозирования кредитного скоринга.

Таблица 1

Характеристики наборов данных

№	Кол-во признаков	Кол-во данных	Особенности
1	200	200.000	Отсутствие категориальных признаков Нормальное распределение данных Нет пропущенных данных
2	370	76.000	Наличие категориальных признаков Наличие аномалий
3	10	33.000	Преобладание категориальных признаков, имеющих множество вариаций значений
4	297	260.000	Наличие категориальных признаков Присутствуют пропущенные данные
5	10	150.000	Наличие категориальных признаков Наличие аномалий

В связи с особенностью многих алгоритмов машинного обучения, большинству наборов данных потребовалась необходимая предобработка данных. В основном это касалось кодирования категориальных признаков и масштабирования данных. Также не обошлось без обработки пустых значений и исправления аномалий.

Для демонстрации приемлемого результата, с помощью функции Grid Search и Keras-Tuner, для каждой модели был осуществлен следующий подбор гиперпараметров:

– LGBM: скорость обучения, количество листьев в каждом дереве, количество итераций бустинга.

– SVM: тип ядра, (для poly степень полиномиальной функции), параметр регуляризации C, гамма, влияющая на поведение ядра.

– DNN: количество слоёв, количество нейронов в каждом слое, размер пакета данных, параметры регуляризации, оптимизаторы.

Для проведения экспериментов использовался компьютер со следующими аппаратными характеристиками:

- CPU: Intel Core i5-3330 3.3 GHz
- GPU: Nvidia Geforce GTX 970 4Gb
- RAM: 16gb DDR3 1333 MHz

2. Критерии сравнения

Для данного исследования были отобраны следующие значимые критерии:

– точность по метрике AUC-ROC:

$$AUC\ ROC = \int_0^1 TPR\, dFPR$$

– где TPR – доля верно предсказанных классов у объектов, относящихся к положительному классу, а FPR – доля неправильно предсказанных классов среди объектов отрицательного класса.

– Ресурсозависимость от времени и оперативной памяти (обучение моделей будет выполняться на CPU).

– Зависимость алгоритмов от объема данных.

3. Результаты экспериментов

Начать обзор результатов эксперимента стоит с самого весомого показателя – точности прогнозирования. В результате настройки моделей, обучения и тестирования на тестовых наборах данных были получены результаты на 100% выборке данных (табл. 2).

Таблица 2

Качество моделей по метрике AUC-ROC

Модель	**1-й набор данных**	**2-й набор данных**	**3-й набор данных**	**4-й набор данных**	**5-й набор данных**
LightGBM	0.88115	0.8101	0.84977	0.96701	0.86812
SVM	0.87616	0.70712	0.87967	0.87831	0.8100
DNN	0.86354	0.81054	0.84361	0.95002	0.86095

Также, в процессе обучения и тестирования была произведена фиксация ресурсозатрат на полных наборах данных. Были получены следующие усредненные результаты:

– LightGBM – ~40 секунд на обучение, ~15 секунд на совершение прогноза. На начало обучения ушло ~700 Mb RAM, но затем быстро снизилось до практически нулевого значения.

– SVM – ~40 минут на обучение, ~15 минут на прогноз. На обучении постоянно использовалось ~200 Mb RAM.

– DNN – ~3 минуты ушло на обучение, ~15 секунд на прогноз. Для обучения потребовалось ~60 Mb RAM.

Исходя из результатов качества модели, можно заметить, что LightGBM почти в каждом наборе данных держал первенство по точности предсказания. Первое, чем данная модель преимущественно выделяется от остальных, это тем, что градиентный бустинг хорошо справляется с аномалиями, которые присутствуют во многих наборах данных, исходя из особенностей деления в деревьях.

Также LightGBM показывает наименьшее время обучения и предсказания. Это можно связать с его особенностями:

– Histogram based splitting – разделение на основе гистограмм. Объединяет непрерывные значения признаков в дискретные ячейки, которые ускоряют процедуру обучения [1].

– GOSS (Gradient Based One Side Sampling) – выборка на основе градиентов. LightGBM усиливает вклад образцов, имеющих малые градиенты, чтобы уделять больше внимания недостаточно обученным экземплярам. Это делает больший акцент на недостаточно обученных экземплярах без значительного изменения распределения данных [2].

– EFB (Exclusive Feature Bundling) – эксклюзивный пакет признаков. EFB объединяет взаимоисключающие ненулевые признаки, чтобы уменьшить сложность обучения [2]. Сложность построения гистограммы в таком случае определяется как: $n_{\text{features}} \cdot n_{\text{bundle}}$

В момент начала обучения градиентный бустинг требует относительно много оперативной памяти, что можно объяснить тем, что LightGBM – алгоритм, основанный на формирование признаков в гистограммы. Из-за того, что гистограммы строятся в начале обучения, происходит двукратное увеличение используемой оперативной памяти, которая затем быстро освобождается.

SVM, в отличие от LightGBM, плохо справляется с аномалиями, так как он может принимать выбросы в качестве опорных данных [3]. Пример его работы с выбросами можно увидеть на рис. 1. Он также неудовлетворительно работает с большим количеством данных и малым количеством признаков. Примечательной является его работа на первом наборе данных, это можно заметить на рис. 2, так как в нем отсутствуют аномалии и присутствует нормальное распределение данных, при котором классическим методам становится проще и быстрее найти

зависимость между признаками и меткой класса (когда речь заходит о задачах классификации) [4]. Благодаря этому, он показал себя лучше, чем LightGBM и нейронная сеть.

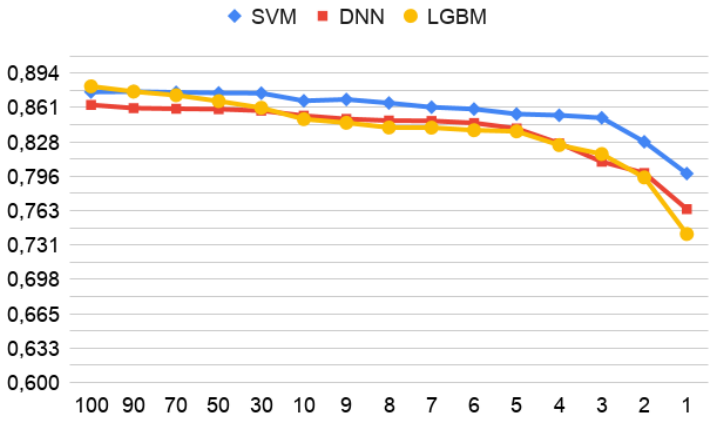


Рис. 1. Зависимость точности моделей от объема в % для пятого набора данных

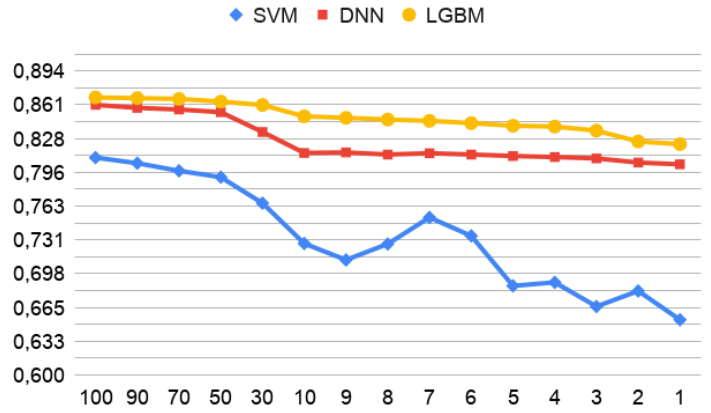


Рис. 2. Зависимость точности моделей от объема в % для первого набора данных

SVM демонстрирует наихудшее время обучения и прогнозирования. Это происходит, из-за особенностей вычисления функции расстояния, сложность которой определяется как – $n_{\text{features}} \cdot n^2_{\text{observations}}$, где $n_{\text{observations}}$ – количество признаков, а n_{features} –

количество элементов в наборе данных. На обучение потребовалось довольно много оперативной памяти, это связано с тем, что он требует, чтобы весь набор данных находился в оперативной памяти одновременно, в отличие от других рассматриваемых здесь алгоритмов. Дополнительной нагрузкой на оперативную память является хранение информации о расстоянии опорных векторов.

Полносвязная нейронная сеть по результатам достаточно близка к LightGBM, но из-за сложности настройки архитектуры и многочисленных гиперпараметров остаётся незначительно позади. В отличие от SVM, нейронная сеть, как и представитель градиентного бустинга, имеет в своем распоряжении алгоритмы оптимизации весовых коэффициенты и работает не с полным набором данных, а делит его на части. В результате этого модель находит “неудачные попытки” и пытается устранить их с помощью алгоритмов оптимизации. Помимо этого скорость обучения нейронной сети напрямую зависит от размера архитектуры сети, размера подвыборки, а также наличия высокопроизводительного GPU, благодаря которому скорость увеличивается примерно в 7 раз, в то время как остальным методам достаточно обычного CPU.

Стоит отметить, что уменьшение исходной обучающей выборки, в основном, до 30% влечет незначительное уменьшение точности (рис. 3).

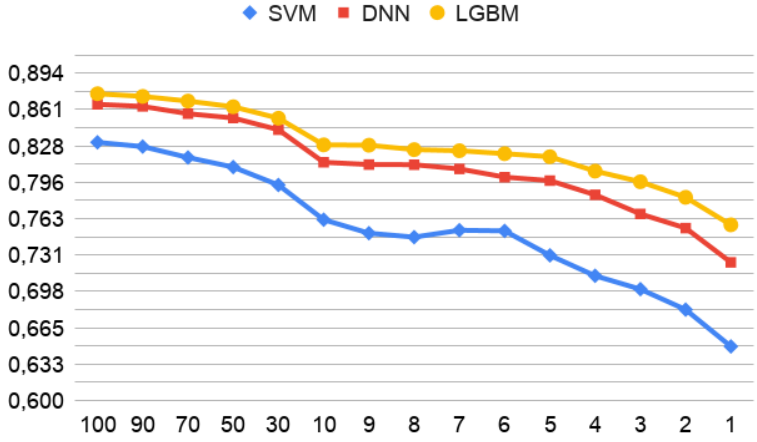


Рис. 3. Зависимость средней точности моделей от объема наборов данных в %

Заключение

По результатам данной работы, можно сделать ряд выводов и рекомендаций:

– Самой быстрой и эффективной моделью оказалась имплементация градиентного бустинга над решающими деревьями LightGBM. Он стабильно себя вел на всех наборах данных и с практически любой подвыборкой, что позволяет добиться удовлетворительного результата в задачах прогнозирования на любых вычислительных мощностях. Перед обучением модели рекомендуется единовременно провести времязатратный подбор гиперпараметров, чтобы добиться максимальной точности. Следует отметить, что при гиперпараметрах по умолчанию, алгоритм часто показывает хорошие результаты, даже по сравнению с настроенной полносвязной нейронной сетью.

– Полносвязная нейронная сеть по точности может не уступать популярным нынче градиентным бустингам, но ей необходима кропотливая и затратная по времени настройка многочисленных параметров, вплоть до построения архитектуры. Рекомендуется ее использовать, если требуется добиться наилучших результатов, например в соревнованиях. Но в реальных задачах с табличными значениями в наборе данных, с целью экономии вычислительных ресурсов и человеко-часов, рекомендуется обратиться к другим моделям.

– Классическая модель машинного обучения, в лице метода опорных векторов, не выдерживает конкуренции на реальных нестерильных данных с выше рассмотренными моделями как по точности, так и по ресурсозатратным характеристикам. Ее рекомендуется использовать на небольших по размеру, очищенных от аномалий, хорошо разделяемых данных, для построения качественных гиперплоскостей. Также перед обучением настоятельно рекомендуется провести подбор параметров регуляризации.

Как уже говорилось ранее, результатами данной работы являются рекомендации касательно использования, а не выявления однозначно лучшего метода. Для любой задачи следует эмпирически подбирать лучшую модель машинного обучения на конкретных данных, но вышеприведенные рекомендации помогут задать начальную точку этим поискам.

Список используемой литературы

1. Li, P. McRank: Learning to Rank Using Multiple Classification and Gradient Boosting / P. Li, Q. Wu, C. Burges // Advances in neural information processing systems. – 2007. – № 20. – P. 897-904.

2. LightGBM: A Highly Efficient Gradient Boosting Decision Tree / G. Ke [et al.] // Advance in neural information processing systems. – 2017. – № 30. – P. 3146-3154.
3. Malossini, A. Detecting Potential Labeling Errors in Microarrays by Data Perturbation / A. Malossini, E. Blanzieri, R. T. Ng // Bioinformatics. – 2006. – № 22. – P. 2114-2121.
4. Transforming Distributions for Machine Learning [Электронный ресурс] : Dummies – Режим доступа : https://www.dummies.com/programming/big-data/data-science/transforming-distributions-machine-learning/

Использование системы компьютерного зрения в сфере нумизматики

А. А. Брянцев

Студент магистрант

В. С. Тарасов

Ассистент

Введение

Последнее время большое развитие получили алгоритмы распознавания изображений. Интеллектуальные системы успешно распознают лица людей, рукописный текст, виды животных и растений, марки автомобилей, а также легко ориентируются в категориях объектов. Тем временем в нумизматике и других видах коллекционирования, существует запрос на быстрое определение и предоставление информации об объекте.

Коллекционирование предполагает изучение и систематизацию знаний об экспонатах, и этим оно отличается от обычного собирательства [1]. Вот почему, для коллекционера важно идентифицировать неизвестный ему экземпляр. Поиск информации может занимать довольно продолжительное время, ведь количество известных видов монет исчисляется в десятках миллионов, и далеко не все монеты содержат достаточные надписи для их классификации.

Целью данной работы является создание мобильного приложения, которое бы позволило осуществлять опознание объекта с помощью встроенной в смартфон камеры. Для решения этой задачи предложен

алгоритм на основе свёрточных нейронных сетей. Набором данных для обучения алгоритма может служить любой онлайн каталог, содержащий фото и описание монет.

1. Обзор существующих решений

Для реализации данного подхода необходимо обратиться к существующим на рынке решениям по распознавания образов в изображении. Сейчас, уже существует довольно много мобильных приложений, которые распознают фотографии для получения некоторой полезной информации о людях или объектах на них.

Google Lens (Объектив) – мобильное приложение Google, в режиме реального времени распознаёт объект на фотографии и предоставляет имеющиеся по нему в интернете сведения. Среди них – определение видов растений и животных, достопримечательностей, поиск товаров.

Алиса – голосовой помощник от Яндекса. Тоже имеет функцию распознавания изображений. На основе загруженного изображения помощник может: узнать знаменитость, авто, породу животного, вид растения, картину и ее автора. Найти товар в Яндекс.Маркет

В приложениях Авто.ру и Авито есть функция определения марки и модели автомобиля по фото. What-Dog.net от Microsoft определяет породу собаки.

Coinscope – Наиболее подходящее приложение позволяющее с достаточной точностью определить монету по фото. Использует следующие каталоги – allnumis.com, catawiki.com, coinscatalog.net. Минусы – реклама, требует подключение к Интернет, использует ограниченное число каталогов.

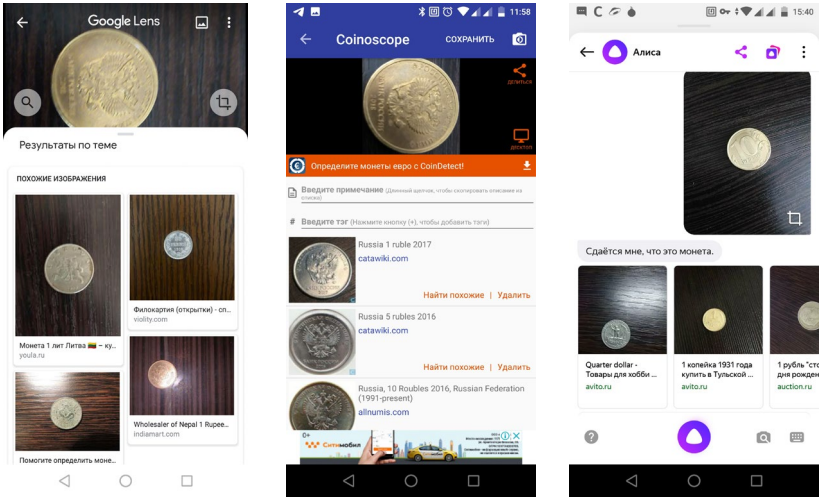
На рис. 1 показано как одну и ту же монету определяют разные системы.

Все вышеперечисленные продукты (за исключением Coinscope) задачу не решают. Они либо специализируются на иной категории (лица, автомобили, животные), либо предлагают универсальные классификаторы (Объектив, Алиса) и будут распознавать монету как категорию, но не конкретный вид. Coinscope же выполняет свою функцию хорошо, но имеет некоторые минусы, описанные выше.

Не секрет, что данные сервисы используют алгоритмы на основе нейронных сетей. Последние несколько лет, свёрточные нейросети в задаче классификации изображений обогнали по эффективности все иные алгоритмы компьютерного зрения и при правильном обучении достигают точности предсказаний до 97% [2].

Однако, при создании свёрточных нейронных сетей встречается ряд трудностей – для каждой задачи необходимо эмпирически подбирать архитектуру нейросети, а также размеры ядер свёртки. Не

менее важным является полнота обучающей выборки и предотвращение переобучения нейросети.



а б в

Рис. 1. Пример распознавания монеты 10 рублей разными системами: а – Google Lens, б – Coinoscope, в – Яндекс Алиса

К техническим ограничениям можно отнести требование достаточно производительной рабочую установки, наиболее оптимальным является видеокарта с технологией CUDA. А процесс обучения может составить несколько суток. Кроме того, неправильно подобранные, громоздкие структуры могут не подходить для работы на мобильных устройствах. К счастью, существует возможность выбрать и построить свою сеть на основе уже натренированной (предобученной) нейросети, в которой сформировались признаки, и переобучить только верхний слой классификатора на своем наборе данных.

2. Датасет

В качестве входных данных для обучения нейросети был взят каталог сайта ucoin.net, в котором на данный момент представлено более 30 млн. монет [3]. Помимо подробного описания и фото аверса и реверса, данный каталог содержит подборку пользовательских фото, снятых разными камерами при различных условиях освещения и состоянии экспоната, что дает дополнительную вариативность для более качественного обучения нейронной сети.

Для загрузки датасета был написан скрипт на языке Python, осуществляющий захват изображений из выбранного раздела сайта и автоматическое сохранение в отдельные каталоги на машине. Название каталога является меткой класса. Для учебной модели были взяты монеты России, Европейского союза и Белоруссии. Всего в выборке оказалось 72 класса и 1588 изображений. Далее, скачанные изображения были поделены на обучающую и тестовую выборки в соотношении примерно 70 к 30.

На рис. 2 показан фрагмент сайта, содержащий страницу описания монеты и фотографии загруженные пользователями.

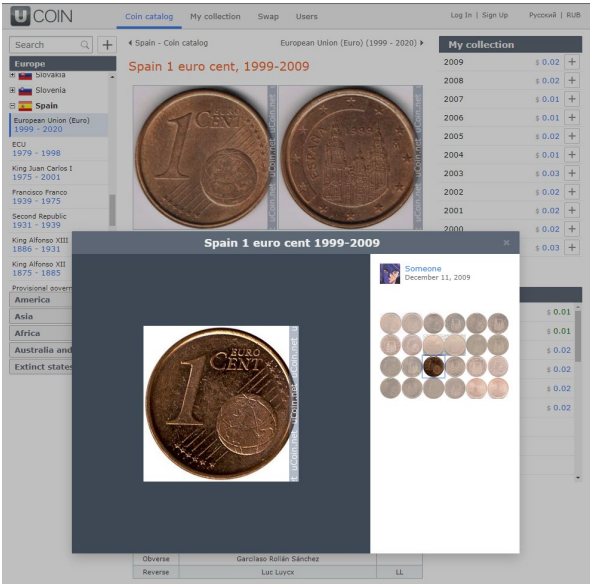


Рис. 2. Пример исходного каталога

3. Обучение

Для построения и обучения модели было решено использовать библиотеку Tensorflow для языка Python. Перед загрузкой данных в нейросеть, необходимо подготовить изображения для обучения.

Алгоритм предварительной подготовки изображений:

1. Загрузить изображение из каталогов и сохранить имя класса
2. Привести все изображения к одному размеру 128 * 128 px
3. Задать случайное вращение и масштабирование, чтобы повысить вариативность.

В данной работе было решено использовать нейросеть MobileNet V2, архитектура которой была специально разработана компанией Google для проведения расчётов непосредственно на мобильных устройствах[4]. Особенностью данной архитектуры является отсутствие слоёв субдискретизации (max pooling). Вместо них для снижения пространственной размерности используется свёртка с параметром сдвига равным 2. Двумя гиперпараметрами архитектуры MobileNet являются α (множитель ширины отвечает за количество каналов в каждом слое) и ρ (множитель глубины или множитель разрешения отвечает за пространственные размеры входных тензоров). Оба параметра позволяют варьировать размеры сети: уменьшая их, мы снижаем точность распознавания, но в то же время увеличиваем скорость работы и уменьшаем потребляемую память.

Эта сеть предварительно обучена на наборе данных ImageNet (база, состоящий из 1,4 млн изображений и 1000 классов) и не включает верхние слои классификации. Далее следует отключить обучение свёрточной базы, мы будем использовать ее в качестве экстрактора объектов. А поверх замороженной сети добавляем и обучаем свой классификатор верхнего уровня.

Для этого мы добавляем всего два слоя. Первый – слой подвыборки, для преобразования объектов в один вектор в 1280 элементов. Усреднение делаем фильтром 5 х 5, используя функцию GlobalAveragePooling2D.

Вторым создаем полносвязный слой (Dense layer), чтобы преобразовать объекты в один прогноз для каждого изображения, в нашем случае содержащий 72 класса.

Итоговая архитектура свёрточной нейросети представлена в листинге.

Листинг

Сводка модели нейронной сети

```
Model: «sequential»
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_128 (Model) (None, 4, 4, 1280)        2257984
_________________________________________________________________
global_average_pooling2d (Gl (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 72)                92232
=================================================================
Total params: 2,350,216
Trainable params: 92,232
Non-trainable params: 2,257,984
```

Обучение проводилось на 1588 изображениях в течении 10 эпох. Точность на обучающей выборке составила 92%. Средствами библиотеки TensorFlow обученная нейросеть была сохранена в формате tensorflowLite для использования в мобильном приложении.

4. Тестирование

Для тестирования было собрано мобильное приложение для системы Андроид, использующее обученную ранее нейросеть.

При наведении камеры смартфона на объект, на экран выводится предположение 3 наиболее вероятных категорий. Тестирование производилось на 10 монетах из тех что были в обучающей выборке. Результат точности определения составил около 80%. На рис. 3 представлен скриншот тестового приложения.

Рис. 3. Тестовое приложение

Заключение

Таким образом, разработан прототип приложения, позволяющий быстро определять монету по фото. Одним из способов дальнейшего повышения производительности является обучение весов (или «точная настройка») верхних слоев предобученной модели наряду с обучением добавляемого классификатора. При этом веса перестроятся от карт общих объектов до объектов, связанных конкретно с нашим набором.

На основе данного подхода можно создать систему распознавания по любому подобному сайту-каталогу коллекционных предметов и при минимальных затратах можно обучить сеть на другом наборе данных.

Список литературы

1. Коллекционирование [Электронный ресурс] : Википедия. Свободная энциклопедия. – Режим доступа : https://ru.wikipedia.org/wiki/коллекционирование
2. Сикорский, О. С. Обзор свёрточных нейронных сетей для задачи классификации изображений / Сикорский О. С. // Новые информационные технологии в автоматизированных системах. – 2017. – С. 73-79.
3. Международный каталог монет [Электронный ресурс]. – Режим доступа : https://ru.ucoin.net
4. MobileNets: Open-Source Models for Efficient On-Device Vision [Электронный ресурс] : Google AI Blog. – Режим доступа : https://ai.googleblog.com/2017/06/mobilenets-open-source-models-for.html

Анализ возможностей применения платформы «1С.ПРЕДПРИЯТИЕ 8» для реализации технологии Data Mining

Д. А. Бутусин

Студент магистрант

В. А. Степанцов

Доцент

Введение

Усиливается конкуренция на рынке программных продуктов фирмы «1С» заставляет ее партнеров-франчайзи искать более эффективные способы по продвижению и продажам программ «1С». Разработана информационная система, которая позволяет помочь франчайзи создать наиболее эффективную стратегию работы с клиентами в направлении продаж [1], внедрения и дальнейшего сопровождения программных продуктов [2].

В качестве среды новой информационной системы была выбрана платформа «1С:Предприятие 8.3», т.к. данная система, являясь предметноориентированной средой разработки, имеет определенные

преимущества по сравнению с предыдущими платформами «1С:Предприятие» [3]. Одной из интереснейшими особенностями новой платформы «1С:Предприятие 8.3» стали встроенные механизмы анализа и прогнозирования данных для решения прикладных задач [4, 5].

1. Механизмы Data Mining встроенные в платформу «1С: Предприятие 8.3»

Механизмы анализа данных представляют собой набор взаимодействующих друг с другом объектов встроенного языка «1С:Предприятие 8». Они позволяют разработчику использовать составные части механизмов в произвольной комбинации в любом прикладном решении [5]. Механизмы позволяют работать как с данными, полученными из информационной базы, так и с данными, полученными из внешнего источника [5]. Общая схема работы механизма анализа и прогнозирования данных представлена на рис. 1. Применяя к исходным данным один из видов анализа, можно получить результат анализа и/или прогноза. Результат анализа представляет собой некую модель поведения данных – он может быть отображен в итоговом документе или сохранен для дальнейшего использования [4, 5]. Дальнейшее использование результата анализа заключается в том, чтобы на его основе создать модель прогноза, позволяющую прогнозировать поведение новых данных в соответствии с имеющейся моделью [4, 5].

В механизме Data Mining реализовано несколько типов анализа и прогнозирования данных:

- общая статистика,
- поиск ассоциаций,
- поиск последовательностей,
- кластерный анализ,
- дерево решений,
- модель прогноза.

2. Тип анализа «Общая статистика»

Представляет собой механизм для сбора информации о данных, находящихся в исследуемой выборке. Этот тип анализа предназначен для предварительного исследования анализируемого источника данных [5]. Анализ показывает ряд характеристик дискретных и непрерывных полей. При выводе отчета в табличный документ заполняются круговые диаграммы для отображения состава полей. На рис. 2 и таблице представлен пример использования данного механизма в созданной информационной системе.

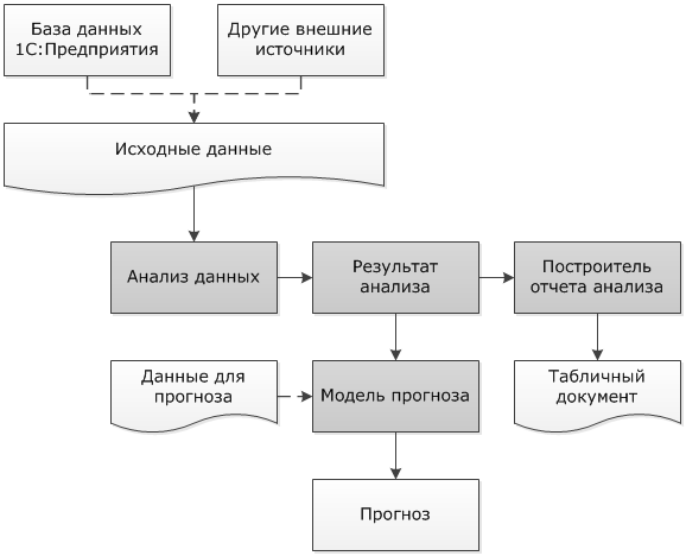
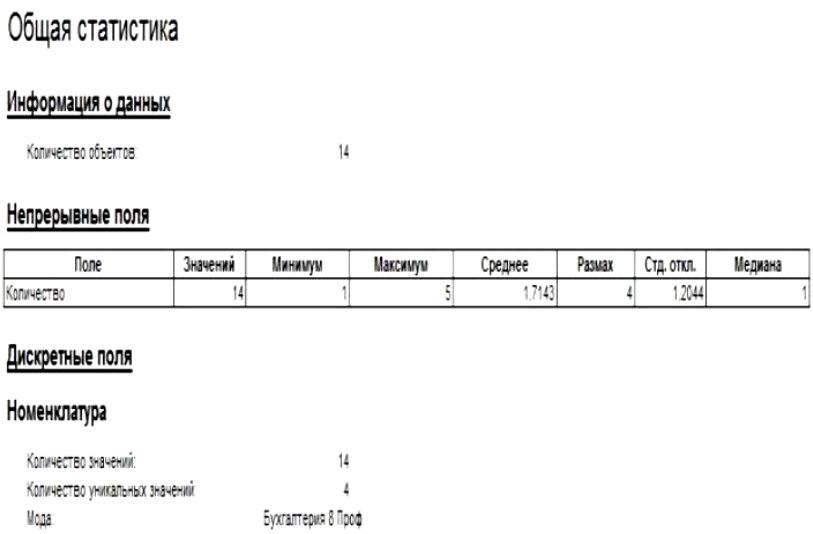


Рис. 1. Общая схема работы механизма анализа и прогнозирования данных

Анализируя полученные данные, можно сделать следующие выводы для анализируемой ситуации (была изучена одна фирма-франчайзи за некоторый период):

1. чаще всего клиенты покупают программный продукт «1С:Бухгалтерия 8 Проф»;
2. в среднем клиенты покупают по 2 программных продукта «1С».

На рис. 3 изображена диаграмма относительных частот по номенклатуре программных продуктов «1С», приобретенных покупателями.



Общая статистика

Информация о данных

Количество объектов 14

Непрерывные поля

Поле	Значений	Минимум	Максимум	Среднее	Размах	Стд. откл.	Медиана
Количество	14	1	5	1.7143	4	1.2044	1

Дискретные поля

Номенклатура

Количество значений: 14
Количество уникальных значений 4
Мода Бухгалтерия 8 Проф

Рис. 2. Пример использования типа анализа «Общая статистика»

Таблица

Исходные данные

Значение	**Частота**	**Относительная частота**	**Накопленная частота**	**Накопленная относительная частота**
Бухгалтерия 8 Проф	6	42,86	6	42,86
Зарплата и управление персоналом 8	3	21,43	9	64,29
Управление торговлей	3	21,43	12	85,71
Розница 2.0	2	14,29	14	100,00

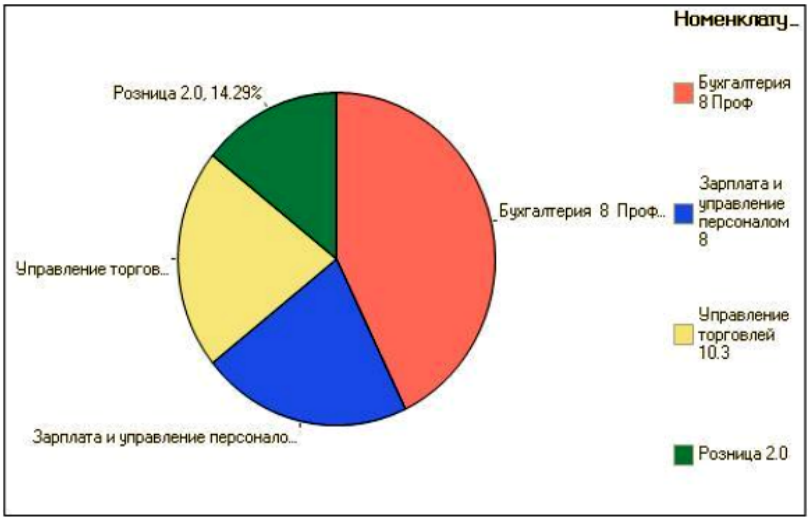


Рис. 3. Относительная частота по номенклатуре

3. Тип анализа «Поиск ассоциаций»

Данный тип анализа осуществляет поиск часто встречаемых вместе групп или значений характеристик, а также производит поиск правил ассоциаций для определения часто приобретаемых вместе товар или услуг. На рис. 4 представлены результаты анализа системы, из которых следует, что клиенты с разными видами деятельности приобретают одинаковые продукты «1С:Управление торговлей 10.3» и «1С:Бухгалтерия 8 Проф». В тоже самое время, второй клиент также купил программный продукт «1С: Зарплата и управление персоналом 8», значит, можно предположить, что первый клиент с вероятностью 11,11 % тоже приобретет данный программный продукт.

4. Тип анализа «Поиск последовательностей»

Этот тип анализа позволяет выявлять в источнике данных последовательные цепочки событий. Позволяет осуществлять поиск по иерархии, что дает возможность отслеживать не только последовательности конкретных событий, но и последовательности родительских групп [4, 5]. На рис. 5 представлен скриншот результата работы программы по поиску последовательностей, показывающий интервалы времени, через которые приобретаются выявленные последовательности программных продуктов «1С».

Результат анализа

Найдено часто встречаемых групп:
Найдено ассоциативных правил:

Часто встречаемые группы

№	Количество случаев	Процент случаев
Состав		
1	1	11,11
Номенклатура = Управление торговлей 10.3; ВидДеятельностиКод = 17.40		
Номенклатура = Бухгалтерия 8 Проф; ВидДеятельностиКод = 17.40		
2	1	11,11
Номенклатура = Бухгалтерия 8 Проф; ВидДеятельностиКод = 52.1		
Номенклатура = Управление торговлей 10.3; ВидДеятельностиКод = 52.1		
Номенклатура = Зарплата и управление персоналом 8; ВидДеятельностиКод = 52.1		

Рис. 4. Результат выполнения типа анализа «Поиск ассоциаций»

Ранее на основе анализа «Поиск ассоциаций» было сделано предположение, что первый клиент также приобретет продукт «1С:Зарплата и управление персоналом 8», как и второй клиент. Теперь на основе анализа «Поиск последовательностей» можно сделать предположение, что первый клиент приобретет программный продукт «1С: Зарплата и управление персоналом 8» в среднем через 1 месяц и 7 дней.

5. Тип «Кластерный анализ»

Кластеризация – выделение из множества объектов одной природы некоторого количества относительно однородных групп – сегментов или кластеров. В основе данного анализа лежит вычисление расстояния между группами, которые и являются кластерами. Определение расстояния между группами производиться различными способами (по разным метрикам) [5].

Результат анализа

Найдено последовательностей: 3

Последовательности

№	Количество случаев	Процент случаев	Средний интервал	Минимальный интервал	Максимальный интервал
Состав					
1	2	40,00			
Номенклатура = Бухгалтерия 8 Проф					
Номенклатура = Зарплата и управление персоналом 8			1 мес. 7 д.	17 д. 5 час.	1 мес. 27 д.
2	1	20,00			
Номенклатура = Управление торговлей 10.3					
Номенклатура = Бухгалтерия 8 Проф			3 д. 19 час.	3 д. 19 час.	3 д. 19 час.
3	1	20,00			
Номенклатура = Бухгалтерия 8 Проф					
Номенклатура = Управление торговлей 10.3			13 д. 10 час.	13 д. 10 час.	13 д. 10 час.

Рис. 5. Результат поиска последовательностей

6. Тип анализа «Дерево решений»

Тип анализа дерево решений позволяет построить иерархическую структуру классифицирующих правил, представленную в виде дерева. Для построения дерева решений необходимо выбрать целевой атрибут, по которому будет строиться классификатор и ряд входных атрибутов, которые будут использоваться для создания правил [4, 5]. Результат работы анализа представляется в виде дерева, каждый узел которого содержит некоторое условие. Для принятия решения, к какому классу следует отнести некий новый объект, необходимо, отвечая на вопросы в узлах, пройти цепочку от корня до листа дерева, переходя к дочерним узлам в случае утвердительного ответа и к соседнему узлу в случае отрицательного. Набор параметров анализа позволяет регулировать точность полученного дерева [5, 6].

7. Модель прогноза

Если рассмотреть задачу прогнозирования в общем, то в ней можно выделить две основные процедуры:

1. Обучение модели на какой-либо выборке.
2. Использование обученной модели для работы с фактическими данными для прогноза.

Используя результаты анализа «Поиска ассоциаций», автоматически строится модель прогноза ассоциаций, из которой

следует, что некий клиент, по аналогии с другими случаями, приобретет предложенный ему товар с определенной степенью вероятности вместе с теми товарами, которые он только что купил. Модель прогноза последовательностей строится, исходя из анализа «Поиск ассоциаций».

Заключение

Использование механизмов Data Mining платформы «1С: Предприятие 8» при подборе программного обеспечения для клиентов фирм франчайзи позволяет искать закономерности в осуществленных продажах программ и строить прогнозные модели, позволяющие автоматически планировать последующие продажи программного обеспечения. Использование данного механизма позволит качественно увеличить процент успешных продаж и внедрений, снизить расходы по данному направлению работы организаций-франчайзи, улучшить качество и скорость обслуживания заказчиков.

Список литературы

1. Лисачева, Е. И. Система поддержки принятия решений организации-франчайзи по подбору программного обеспечения для покупателей / Е. И. Лисачева, А. Н. Важдаев // Ползуновский вестник. – 2013. – №. 2. – С. 224-228. [Электронный ресурс]. – Режим доступа : https://1c.ru/rus/partners/training/edu/theses/ ?y=2014&s=84&t=2309

2. «1С: Франчайзинг» [Электронный ресурс]. – Режим доступа : http://www.1c.ru/rus/firm1c/franch.htm

3. Нуралиев, С. Г. Является ли «1С:Предприятие» средством разработки? / С. Г. Нуралиев // [Электронный ресурс]. – Режим доступа : http://www.lawlinks.ru /viewdata.id=132536

4. Радченко, М. Г. 1С:Предприятие 8.2. Практическое пособие разработчика. Примеры и типовые примеры / М. Г. Радченко, Е. Ю. Хрусталева. // М. : ООО «1С-Паблишинг», 2009. – С. 874. [Электронный ресурс]. – Режим доступа : https://www.mann-ivanov-ferber.ru/books/1c-pablishing/posobie/

5. Гончаров, Д.И. Решение специализированных прикладных задач в «1С:Предприятии 8.2» / Д. И. Гончаров, Е. Ю. Хрусталева. – М. : «1С-Паблишинг». – 2012. – 300 с. [Электронный ресурс]. – Режим доступа : http://info1c.pro/knigi/?ELEMENT_ID=56

6. Лисачева, Е. И. Оценка и подбор программных продуктов для клиентов / Е. И. Лисачева, А. Н. Важдаев. // Сборник научных трудов 14-й МНПК «Новые информационные технологии в образовании». – 2014. – С. 28-29. – [Электронный ресурс]. – Режим доступа : https://1c.ru/rus/partners/training/edu/theses/ ?y=2014&s=84&t=2309

Построение модели выбора поставщика и распределения ресурсов в условиях противоречивости целевых установок

А. М. Воронина

Студент магистрант

М. Г. Матвеев

Профессор

Введение

Проблема выбора поставщика на предприятие является одной из самых стратегически важных для функционирования любой компании. Чаще всего в современных условиях данная проблема возникает у компаний в связи со следующими условиями: введение на предприятии нового проекта, требующего закупку оборудования; замена устаревшего оборудования на более новое, иначе – амортизация оборудования; реинжиниринг компании или отдельного предприятия. На современном рынке существует большое количество предприятий-поставщиков, которые реализуют ресурсы схожие по своим показателям, востребованности и свойствам. Помимо этого, каждое предприятие-поставщик обладает собственной политикой ведения коммерческих дел. Все это делает процедуру выбора потенциального поставщика ресурсов на предприятие актуальной и востребованной на текущий момент времени.

В данной работе рассмотрен подход к решению задачи выбора ресурсов и поставщиков, а также построена модель выбора поставщика, обеспечивающая одновременное удовлетворение технических и коммерческих требований к закупаемому ресурсу.

1. Термины и определения

Закупочная организация – организационная единица, которой поручается поиск и отбор потенциальных предприятий-поставщиков, тем самым делая её ответственной за все закупочные операции компании. Она заготавливает товары и ведет переговоры об условиях закупочной цены с поставщиками [1].

Технические требования – представляют собой перечень технических характеристик ресурса с обязательным указанием допустимых значений, представленных в виде диапазона. Примером технических характеристик могут служить производительность, срок

службы, а так же индивидуальные характеристики конкретного типа оборудования (скорость, ёмкость, длина, частота и т.д.)

Коммерческие требования – представляют собой перечень коммерческих характеристик с указанием диапазона допустимых значений. Примером коммерческих характеристик могут служить цена, сроки поставки, условия поставки и т. д.

Закупочная организация получает и обрабатывает заявки на приобретение необходимого количества определенного типа ресурса от компаний-заказчиков, с указаниями допустимых диапазонов требуемых значений технических характеристик. Далее закупочная организация формирует собственный перечень коммерческих характеристик оборудования, который напрямую будет зависеть от выдвинутых ранее заказчиком технических требований. Выдвигаемым заказчиком техническим требованиям, могут соответствовать несколько типов ресурсов. Ресурсы, имеющие одинаковые, характерные для них идентичные признаки и обладающие одинаковым назначением называются однородными.

Задача закупочной организации сводится к формированию портфеля закупок с обязательным соблюдением баланса между техническими и коммерческими требованиями к типам закупаемых ресурсов. Зависимость технических и коммерческих требований друг от друга является пропорциональной, т.е. постановка более высоких технических характеристик влечет за собой рост значений коммерческих характеристик. Иными словами, выбор поставщика осуществляется в противоречивых условиях, с различными целевыми функциями, что влечет за собой необходимость решения многокритериальной задачи оптимизации.

2. Формализация технических и коммерческих требований

Важным правилом технической политики является формирование технических требований обезличено, без привязки к конкретным типам оборудования, что позволяет осуществлять конкурсные процедуры закупки более гибко, с максимальным увеличением ассортимента потенциальных предложений [2-3]. Автоматизация процессов закупки должна основываться на выборе такого конкретного оборудования в рамках закупочного лота, характеристики которого максимально соответствуют техническим требованиям. Технические требования должны задаваться в форме списка технических характеристик с указанием диапазона допустимых значений. Помимо задания диапазонов допустимых значений необходимо также сформировать функции, которые будут непосредственно отражать соответствие конкретного значения оборудования внутри заданного интервала. Такие

функции являются функциями принадлежности. Сравнение реальных показателей ресурса происходит с функцией принадлежности, в результате чего появляется возможность определить степень соответствия выбранного ресурса по каждой из технических характеристик. Степень соответствия отдельных характеристик техническим требованиям можем представить в виде $a_1, a_1 \in [0,1]$.

Полученные по каждой из характеристик значения a_1 записываются в вектор соответствия $\vec{a}_i = (a_1, ..., a_l)$ в порядке следования критериев (характеристик).

Агрегация вектора соответствия будет выполняться по формуле среднего арифметического, т.к. в случае применения агрегирования по минимальному или максимальному критерию, значительное изменение одной из частных оценок может ни как ни отразиться на результате выполнения операции, что приводит к нецелесообразности использования данного метода. Таким образом, агрегирование векторных показателей осуществляется по формуле

$$A_i = \frac{1}{n} \cdot \sum_{i=1}^{n} a_i \,. \qquad (1)$$

Проагрегированные показатели A_i записываются в матрицу соответствия η_{jk}. Формирование матрицы соответствия η_{jk} происходит на основе множества однородных ресурсов $J = \{j\}$, среди которых $j = 1, ..., m$ – тип однородного ресурса, а так же на основе множества заказчиков $K = \{k\}$, среди которых $k = 1, ..., d$ – заказчик с уникальным именем. Каждому соответствию $jk \in J \times K$ присваивается значение проагрегированного вектора $\vec{a}_i$, которое будет равно A_i. Именно это значение присваивается в матрицу η_{jk}.

Потребное количество ресурса j заказчику обозначим в следующей формуле

$$\upsilon_k = \sum_{j=1}^{m} y_{jk} \,. \qquad (2)$$

и включим это значение, вместе со значением соответствия в виде матрицы η_{jk} в заявку заказчика и обозначим:

$$z_k = \{(\eta_{jk}); \upsilon_k\} \,. \qquad (3)$$

Аналогичным образом задаем формализацию коммерческих требований. Степень соответствия отдельных характеристик техническим требованиям можем представить в виде $\beta_1, \beta_1 \in [0,1]$. Полученные по каждой из характеристик значения β_1 записываются в вектор соответствия $\vec{\beta}_i = (\beta_1, ..., \beta_l)$ в порядке следования критериев (характеристик). Агрегирование векторных показателей осуществляется по формуле

$$B_i = \frac{1}{n} \sum_{i=1}^{n} \beta_i \ . \tag{4}$$

Проагрегированные показатели B_i записываются в матрицу соответствия μ_{ij}. Формирование матрицы соответствия μ_{ij} происходит на основе множества однородных ресурсов $J = \{j\}$, среди которых $j = 1, ..., m$ - тип однородного ресурса, а так же на основе множества ценовых предложений поставщиков $I = \{i\}$, среди которых $i = 1, ..., n$ – конкретное предложение поставщика. Каждому соответствию $ij \in I \times J$ присваивается значение проагрегированного вектора $\vec{\beta}_i$, которое будет равно B_i. Именно это значение присваивается в матрицу μ_{ij}.

3. Формирование функций критериев и ограничений

Технические требования можно сформулировать следующим образом:

$$\sum_{j=1}^{m} \sum_{k=1}^{d} \eta_{jk} y_{jk} \to \max \ . \tag{5}$$

где η_{jk} – значения проагрегированных векторов соответствия реального оборудования выдвигаемым техническим требованиям, записанные в матрицу, а y_{jk} – количество однородного ресурса, назначенное определенному заказчику.

На данные технические требования накладываются следующие ограничения:

$$\sum_{j} y_{jk} = \upsilon_k ; \forall k \ . \tag{6}$$

$$r_j = \sum_{k} y_{jk} ; \forall j \ . \tag{7}$$

где υ_k - потребное определенному заказчику количество ресурса, а r_j - количество каждого типа однородного ресурса, которое необходимо закупить. Первое ограничение означает, что каждое предприятие – заказчик должно получить количество ресурсов в требуемом количестве. Второе накладываемое ограничение означает, что количество однородного ресурса, требуемое совокупностью заказчиков, одного типа будет полностью распределено между предприятиями – заказчиками. То есть будет закуплено не меньше и не больше требуемого количества.

Коммерческие требования можно сформулировать следующим образом

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\mu_{ij}x_{ij} \to \max . \qquad (8)$$

где μ_{ij} – значения проагрегированных векторов соответствия реальных предложений поставщиков и выдвинутых коммерческих требований, записанные в матрицу соответствия, а x_{ij} - множество однородного ресурса, закупаемого у определенного поставщика.

На данные критерии так же накладывается ряд ограничений:

$$\sum_{i} x_{ij} = r_j ; \forall j . \qquad (9)$$

где r_j – количество каждого типа однородного ресурса, которое необходимо закупить, а x_{ij} – множество j ресурса закупаемого у i поставщика. Данное ограничение означает, что все однородные ресурсы, относящиеся к одному типу и предлагаемые поставщиком, должны быть приобретены.

В виду того, что критерии, созданные по техническим и коммерческим требованиям, зависят от переменной r_j, которая в свою очередь зависит от переменных y_{jk} и x_{ij}, возникает противоречие. Решая отдельно задачи нахождения технических и коммерческих критериев, будут получены различные r_j. Это значит, что приобретаться у поставщиков будет одно количество ресурсов, а распределяться между поставщиками будет другое количество ресурсов. Таким образом, возникает многоцелевая задача оптимизации по двум критериям. В качестве решения данной проблемы, необходимо привести многоцелевую задачу к скалярному виду, что представляется возможным с использованием аддитивной свертки критериев, т.к. все функции критериев являются линейными и выпуклыми.

4. Аддитивная свертка критериев

Аддитивная (линейная) свертка применяется в условиях, когда необходимо принять решение, выбрав один из вариантов в качестве оптимального, при этом существуют критерии эффективности решения [4]. Линейная свертка требует знания оценки вариантов по каждому из критериев и знания о весе (приоритете) критерия для решаемой задачи.

Воспользовавшись формулами аддитивной свертки и приняв весовые коэффициенты обоих критериев равными между собой, а соответственно, исходя из условия, что сумма всех весов должна быть равна единице, принимаем веса критериев равными 0,5. В таком случае свертка критериев будет выглядеть следующим образом.

$$0{,}5\sum_i\sum_j \mu_{ij}x_{ij} + 0{,}5\sum_j\sum_k \eta_{jk}y_{jk} \to \max . \tag{10}$$

Однако требование равенства весов может не соблюдаться, если при решении поставленной задачи принимается условие, что один из критериев является более значимым, чем другой. Тогда общий вид модели выбора поставщика на основе технических и коммерческих требований будет выглядеть следующим образом,

$$\lambda\sum_i\sum_j \mu_{ij}x_{ij} + (1-\lambda)\sum_j\sum_k \eta_{jk}y_{jk} \to \max . \tag{11}$$

где λ – значение веса первого критерия (в данном случае коммерческого), $(1-\lambda)$ - значение веса второго критерия (в данном случае технического), $\sum_j\sum_k \eta_{jk}y_{jk}$ - заданные технические требования, а $\sum_i\sum_j \mu_{ij}x_{ij}$ – заданные коммерческие требования.

Построенная модель будет иметь следующие ограничения:

$$\sum_j y_{jk} = \upsilon_k ; \forall k . \tag{12}$$

$$\sum_i x_{ij} = a_i ; \forall i . \tag{13}$$

$$\sum_j x_{ij} = \sum_j y_{jk} ; \forall j . \tag{14}$$

$$x_{ij} \geq 0; \forall i, j . \tag{15}$$

$$y_{jk} \geq 0; \forall j, k . \tag{16}$$

Первое ограничение означает, что каждое предприятие – заказчик должно получить количество ресурсов в требуемом количестве. Второе ограничение означает, что количество закупаемого у конкретного

поставщика ресурса должно быть равно количеству предлагаемого поставщиком ресурса. Следующее, накладываемое на модель ограничение, означает, что количество приобретаемого и распределяемого ресурса должно быть равно между собой. Таким образом, устраняется противоречие, возникнувшее при решении отдельно поставленных задач.

Полученная модель относится к классу задач линейного целочисленного программирования, которая будет решена, основываясь на методе ветвей и границ.

5. Методы отсечения в целочисленном линейном программировании

Задачей целочисленного линейного программирования (ЗЦЛП) называется задача максимизации или минимизации линейной функции при линейных ограничениях, в которой на значения всех или части переменных наложено условие целочисленности [5]. Метод ветвей и границ имеет ряд преимуществ перед алгоритмами, использующими исключительно отсекающие плоскости. Одно из преимуществ — алгоритм можно завершить рано, как только хотя бы одно допустимое целочисленное решение найдено, хотя и не оптимальное. Кроме того, решение ослабленной линейной задачи может быть использовано для оценки, насколько далеко полученное от оптимального решения. Наконец, методы ветвей и границ можно использовать, чтобы получить несколько оптимальных решений.

Обобщенная идея метода ветвей и границ может быть описана на примере поиска минимального значения функции $f(x)$, на множестве x допустимых значений. Особенностью метода ветвей и границ является необходимость проведения двух процедур: ветвления и нахождения оценок (границ). Первая процедура состоит в разложении множества допустимых значений переменной x на подмножества. Данную процедуру так же можно рекурсивно применять к подмножествам. Полученные в результате подобного разбиения подмножества образуют дерево, которое носит название дерево ветвей и границ (дерево поиска). Узлами такого дерева являются построенные подмножества из множества значений переменной x.

Поиск верхних и нижних границ, определяемых на подмножестве, необходим в процедуре нахождения оценок. Метод ветвей и границ базируется на идее о том, что если нижняя граница значений функции на подмножестве A дерева поиска будет больше, чем верхняя граница на каком-либо ранее просмотренном подмножестве B, то A может быть исключено из дальнейшего рассмотрения (правило отсева).

В случае если нижняя граница для узла дерева совпадает с верхней границей, то это значение является минимумом функции и достигается на соответствующем подмножестве.

Заключение

Данная статья посвящена созданию модели, целью которой является отбор поставщиков и распределение ресурсов с учетом противоречивости технических и коммерческих требований и учетом распределения приоритетов между этими требованиями. Также были формализованы и приведены к общему виду технические и коммерческие требования, применительно к задаче выбора поставщика.

Список литературы

1. SAP Help Portal Библиотека SAP. Базовые принципы / SAP Help Portal : [Электронный ресурс]. – Режим доступа : http://help.sap.com/saphelp_afs64/helpdata/ru/
2. Еремина, Е. А. Нечеткая модель выбора поставщика / Е. А. Еремина // Молодой ученый. – 2011. – №11. – Т.1. – С. 120-122.
3. Андрейчиков А. В. Анализ, синтез принятие решений : Учебник для студентов высших учебных заведений / А. В. Андрейчиков, О. Н. Андрейчикова. – М. : Финансы и статистика, 2000. – 200 с.
4. Ногин, В. Д. Линейная свертка критериев в многокритериальной оптимизации / В. Д. Ногин // Искусственный интеллект и принятие решений. – 2014. – №4. – С. 73-82.
5. Шевченко, В. Н. Линейное и целочисленное линейное программирование / В. Н. Шевченко, Н. Ю. Золотых. – Нижний Новгород : Издательство Нижегородского госуниверситета им. Н. И. Лобачевского, 2004. – 154 с.

Сравнительный анализ технологий программирования графических процессоров

Д. С. Галиулин

Студент магистрант

С. В. Борзунов

Доцент

Введение

В современном мире, в эпоху цифровых технологий, где важна быстрая обработка больших объемов данных, возникает потребность в высокопроизводительных вычислениях. Одним из вариантов решения данной проблемы является внедрение параллелизма для вычислительных систем и использование мощностей графических процессоров (GPU).

Типичный графический процессор содержит тысячи ядер, которые могут независимо от других выполнять однотипные операции. Расчеты на GPU получили большое распространение в таких технологиях как нейронные сети, обработка изображений, а также в математических и финансовых расчетах, где требуется оперировать большим объемом вычислительных данных.

Целью работы является сравнение двух технологий программирования на графических процессорах OpenCL и CUDA, которые зарекомендовали себя как высокопроизводительные технологии, решающие большой спектр задач и позволяющие достигать поставленных целей.

1. CUDA

CUDA – платформа, которая может значительно повысить производительность вычислений за счет использования графических процессоров Nvidia. Представляет собой C-подобный язык программирования с компилятором и вычислительными библиотеками на GPU.

CUDA использует большое количество отдельных потоков для расчетов. Все они группируются в иерархию – *grid / block / thread*. *Grid* является верхним уровнем в модели – отвечает ядру и объединяет все потоки, которые выполняет данное ядро. *Grid* – одномерный или двумерный массив блоков (*block*). Каждый блок *block* представляет

собой полностью независимый набор скоординированных между собой потоков. Каждый поток обладает своей памятью (*local),* недоступной другим потокам. Внутри блока потоки имеют разделяемую память (*shared*) и синхронное исполнение. Потоки из разных блоков не могут взаимодействовать, но всем потокам доступна общая глобальная память.

Программа на CUDA (как и в OpenCL) разделяется на две части: первая часть – управляющая, вторая – вычислительная. В роли управляющего устройства (*host*) выступает центральный процессор (CPU), а в качестве вычислительного устройства (*device*) используется GPU.

На центральном процессоре (хосте) выполняются только последовательные части алгоритма программы, подготовка и копирование данных на устройство, задание параметров для ядра и его запуск. Параллельные части алгоритма оформляются в ядра, которые выполняются на большом количестве потоков на устройстве [1].

2. OpenCL

OpenCL – это открытый API (англ. application programming interface, программный интерфейс приложения) – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой, для параллельных вычислений, разработанный для того, чтобы графические процессоры и другие сопроцессоры могли работать в тандеме с центральным процессором, обеспечивая дополнительную вычислительную мощность.

В OpenCL, как и в CUDA, используется иерархия, основанная на потоках (*work-item*). Потоки объединены в рабочие группы (*work-group*). Каждый поток имеет свою память (*private*), а также доступ к разделяемой памяти группы (*local*) и глобальной памяти устройства (*global*).

В программе на OpenCL вычислительное устройство выбирается из списка платформ и их устройств. Обычно используется GPU, но также в качестве устройства может быть выбран CPU.

Общая структура программы является более сложной, чем аналогичная программа на CUDA, поскольку необходимо выполнять дополнительные действия по конфигурированию среды выполнения и подготовке кода для устройства.

На центральном процессоре первым шагом необходимо получить информацию о платформах, выбрать устройства и подготовить для них контекст. Следующий этап – создать ядро: получить и скомпилировать текст программы для выполнения из исходного кода. Для запуска полученного ядра необходимо выделить память и подготовить данные

для обработки, создать очередь команд устройства и назначить параметры выполнения ядра. По завершении всех работ требуется освободить ресурсы: контекст, очередь программы, выделенную под данные память [2, 3].

3. Задача перемножения матриц

Задача заключается в нахождении матрицы $C[M * K]$, полученной путем перемножения двух заданных прямоугольных матриц $A[M * N]$ и $B[N * K]$ с вещественными коэффициентами.

4. Реализация ядра на CUDA

Первый вариант реализации основан на использовании глобальной памяти (программная реализация приведена в листинге 1). Схема работы алгоритма заключается в следующем:

1. Матрицы A и B разбиваются на блоки заданного размера.
2. В каждом блоке функционируют потоки, количество которых зависит от размера блока. Координаты каждого потока определяются координатами внутри блока.
3. Каждый поток читает одну строку из матрицы A, один столбец из матрицы B и вычисляет соответствующий элемент матрицы C.

Листинг 1

```
__global__ void matrix_multiply_kernel(double* a, double* b,
double* c, int m, int n, int k) {
    int row = blockIdx.y * blockDim.y + threadIdx.y;
    int col = blockIdx.x * blockDim.x + threadIdx.x;
    double sum = 0.0;
    if (col < k && row < m){
        for (int i = 0; i < n; i++)
            sum += a[row * n + i] * b[i * k + col];
        c[row * k + col] = sum;
    }
}
```

Ключевое слово *__global__* является входной точкой в функцию для выполнения кода на устройстве GPU. Переменная *blockIdx* определяет положение/индекс блока внутри сетки, *blockDim* – размер блока, *threadIdx* – индекс потока внутри блока.

Принято считать, что глобальная память работает довольно медленно, поэтому, для ускорения вычислений и работы потоков, рассмотрим второй вариант реализации алгоритма перемножения матриц с «разделяемой» памятью.

Основные особенности в работе алгоритма с разделяемой памятью между потоками (см. листинг 2):

1. Матрицы A и B разбиваются на непересекающиеся блоки, которые загружаются в разделяемую память, доступ к которой намного быстрее, чем к глобальной.

2. Для уменьшения числа доступов к глобальной памяти каждый поток загружает только часть матриц A и B в разделяемую память.

3. Каждый поток отвечает за подсчет суммы произведений строки на столбец подматриц из A и B и сохраняет только одно значение в C.

4. Когда каждый поток подсчитал сумму блока, загружается следующая порция блоков подматриц A и B.

5. Вычисления завершаются, когда все подблоки матриц A и B были обработаны.

Листинг 2

```
__global__ void matrix_multiply (double* C, double* A, double*
B, int wA, int hA, int wB){
   int bx = blockIdx.x; //Координата блока по Оси X
   int by = blockIdx.y; //Координата блока по Оси Y
   int tx = threadIdx.x; //Координата потока внутри блока
   int ty = threadIdx.y; //Координата потока внутри блока
   int aBegin = wA * BLOCK_SIZE * by; //Первая подматрица из A
   int aEnd = aBegin + wA - 1; //Последняя подматрица из A
   int aStep = BLOCK_SIZE; //Шаг
   int bBegin = BLOCK_SIZE * bx; //Первая подматрица из B
   int bStep = BLOCK_SIZE * wB; //Шаг
   double Csub = 0;
   for (int a = aBegin, b = bBegin; a <= aEnd; a += aStep, b
+= bStep) {
      //Объявление переменных для использования общей памяти
      __shared__ double As[BLOCK_SIZE][BLOCK_SIZE];
      __shared__ double Bs[BLOCK_SIZE][BLOCK_SIZE];
      //Каждый поток загружает только по 1 элементу в блоке
      As[ty][tx] = A[a + wA * ty + tx];
      Bs[ty][tx] = B[b + wB * ty + tx];
      //Синхронизация выполнения работы всех потоков
      __syncthreads();
     //Каждый поток вычисляет только один элемент матрицы C
      for (int k = 0; k < BLOCK_SIZE; ++k) {
         Csub += As[ty][k] * Bs[k][tx];
      }
     __syncthreads();//Ожидание завершения работы всех потоков
   }
   C[wB * BLOCK_SIZE * by + BLOCK_SIZE * bx + wB * ty + tx] =
Csub; //результат в глобальную память
}
```

5. Реализация ядра на OpenCL

Основным отличием реализации программы на OpenCL от реализации на CUDA являются используемые директивы. Принцип и подход к вычислению остается теми же самыми.

Идентификатор *__kernel* используется для указания функции, которая будет вызываться на хосте и выполняться на устройстве (*__global__* в CUDA). С помощью функции *get_group_id* получаем номер *work-group* (*block* в CUDA), а идентификатор *work-item* (*thread* в CUDA), который сейчас выполняется, получаем с помощью *get_local_id.* Для синхронизации работы всех поток используется функция *barrier*.

6. Гравитационная задача

Начальное положение N точечных тел. Каждое тело характеризуется массой m и начальной скоростью v. Предполагается, что на тела не действуют никакие силы, кроме гравитационных. Под действием гравитации тела изменяют свое положение, а также скорость. Необходимо смоделировать взаимодействие тел друг с другом через заданный промежуток времени и определить конечные координаты.

Математическая модель задачи использует следующие формулы:

1. Закон всемирного тяготения:

$$\vec{F} = G\frac{m_1 \times m_2}{r^2}. \quad (1)$$

2. Правило сложения сил: если на тело действуют несколько сил, то их можно заменить одной силой, равной векторной сумме всех исходных сил.

3. Второй закон Ньютона:

$$\vec{F} = m\vec{a}. \quad (2)$$

Взаимодействие тел моделируется пошагово с помощью дискретных отрезков времени фиксированной длительности dt. На каждом шаге вычисляется сила, действующая на каждое тело в начальный момент времени, по найденной силе определяется ускорение.

Для простоты предполагается, что все тела расположены в одной плоскости. Положение каждого тела определяется двумя координатами (x, y) [4].

7. Реализация ядра на CUDA

В качестве ограничения будем считать гравитационную постоянную равной 10, шаг по времени равным 0.1 секунды. Для случаев, когда взаимодействующие тела располагаются близко друг от друга, вводим максимальную силу ($F_{\max}$), равную 1.

Основной особенностью реализации гравитационной задачи является то, что вычисление новых координат и скоростей тел разбито на два этапа:

1. Вычисление сил, действующих на частицу, происходит на устройстве GPU. По итогу полученные данные возвращаются на CPU.

2. Координаты и скорости вычисляются на CPU на основе полученных с GPU сил.

Первый вариант реализации (листинг 3) рассмотрим с использованием глобальной памяти устройства. При вычислении каждый поток вычисляет силы, действующие со стороны других частиц на одну конкретную частицу.

Листинг 3

```
__global__ void forceGpu(double4* p, double4* f, double* m) {
 int i = blockDim.x * blockIdx.x + threadIdx.x;
 if (i < N){
  for (int j = 0; j < N; j++){
   if (i == j) continue;

   double dx = p[j].x – p[i].x;
   double dy = p[j].y – p[i].y;
   double r_2 = 1 / (dx * dx + dy * dy);
   double r_1 = sqrtf(r_2);
   double fabs = gravity * m[i] * m[j] * r_2;
   if (fabs > Fmax) fabs = Fmax;

   f[i].x = f[i].x + fabs * dx * r_1;
   f[i].y = f[i].y + fabs * dy * r_1;
  }
 }
}
```

Номер частицы, для которой текущий поток будет вычислять силы, определяется на основе значений: *blockIdx* – положение блока внутри сетки, *blockDim* – размер блока, *threadIdx* – индекс потока внутри блока.

Второй вариант вычисления сил основывается на использовании разделяемой памяти потоками внутри блока (листинг 4). Все тела разделяют на группы. Каждый поток вычисляет общую силу, которая действует на тело со стороны других тел, внутри блока до тех пор, пока все блоки не будут обработаны.

Листинг 4

```
__global__ void forceGpu(double4 p, double4 f, double* m) {
   int i = blockDim.x * blockIdx.x + threadIdx.x;
   //Переменные в разделяемой памяти устройства
   _shared_ double sm[BLOCK_SIZE]; //массы частиц
   _shared_ double4 spos[BLOCK_SIZE]; //координаты частиц
```

```
    for (int tile = 0; tile < gridDim.x; tile++) {
      //каждый поток получает массы и координаты для 1 частицы
       spos[threadIdx.x] = p[tile * blockDim.x + threadIdx.x];
       sm[threadIdx.x] = m[tile * blockDim.x + threadIdx.x];
       __syncthreads();//Синхронизация выполнения всех потоков

       for (int j = 0; j < BLOCK_SIZE; j++) {
          int gj = tile * BLOCK_SIZE + j;
          if (i == gj || gj >= N || i >= N){ continue; }
          … //вычисление dx, dy, r_2, r_1, fx, fy
       }
       __syncthreads();//Ожидание для перехода к новому блоку
    }
}
```

8. Реализация ядра на OpenCL

Как и в случае с задачей перемножения матриц, реализация двух вариантов программы на OpenCL полностью совпадает с реализацией на CUDA, за исключением используемых конструкций и директив.

Для получения текущего номера тела используются специальные функции: *get_local_size* – для определения размера рабочей группы, *get_group_id* – номера группы и *get_local_id* – номера рабочего элемента в группе.

В реализации гравитационной задачи с общей памятью внутри рабочей группы используются директива *__local* – определение локальной памяти группы и функция *barrier()* – для синхронизации потоков внутри группы для продолжения обработки следующего блока.

9. Сравнение результатов работы

В результате ряда испытаний с постоянным увеличением размера матриц технологии OpenCL и CUDA показали практически идентичные результаты в реализации с глобальной памятью. Вариант с разделяемой памятью на CUDA дал немного лучший результат по времени (см. рис. 1).

В гравитационной задаче результаты вычислений показали небольшой разброс. Реализация с глобальной памятью с использованием технологии OpenCL выполнялась немного медленнее, чем на CUDA. Реализация с разделяемой памятью на OpenCL показала лучший результат, чем на CUDA (см. рис. 2).

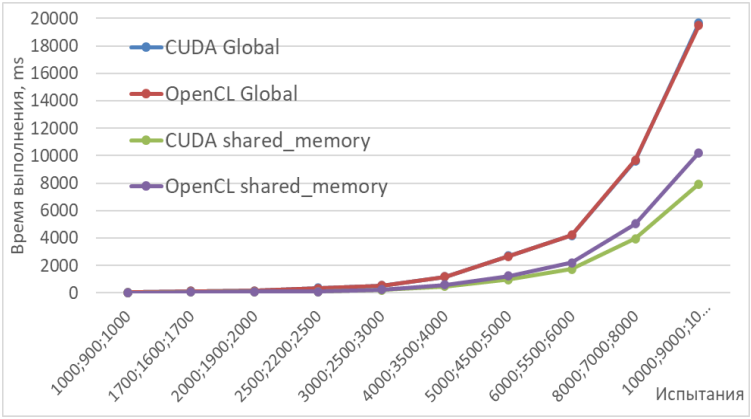


Рис. 1. Время работы программ перемножения матриц

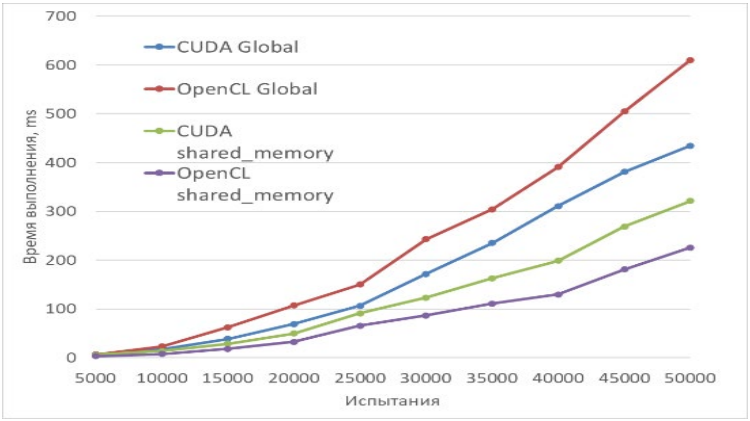


Рис. 2. Время работы программ решения гравитационной задачи

Заключение

Данная статья посвящена сравнению популярных технологий программирования графических процессоров CUDA и OpenCL. В ходе работы был реализован программный код для рассматриваемых задач и оптимизирован для получения наилучшего результата. Проведены циклы запусков и собраны метрики, которые представлены на графике.

В итоге можно сделать вывод, что указанные технологии очень похожи по подходам и организации и по результатам проведенных испытаний показали соизмеримые результаты на исследуемых задачах.

Список литературы

1. NVIDIA CUDA Programming Guide [Электронный ресурс] / NVIDIA Corporation. – Режим доступа : http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci360/NVIDIA_CUDA_Programming_Guide_2.3.pdf/
2. Tomson J. An Introduction to the OpenCL Programming Model [Электронный ресурс] / J. Tompson, K. Schlachter. – Режим доступа: https://cims.nyu.edu/~schlacht/OpenCLModel.pdf/
3. Muhchi A. OpenCL Programming Guide [Электронный ресурс] / A. Munshi [et al]. – Режим доступа : http://asu-cs.donntu.org/sites/default/files/images/doc/opencl.programming.guide.pdf/
4. Параллельные методы решения гравитационной задачи n тел [Электронный ресурс] : база данных. – Режим доступа : http://edu.mmcs.sfedu.ru/pluginfile.php/5634/mod_resource/content/5/OMPMPIGravit2019.pdf/

Разложение в ряд по системе целочисленных сдвигов функции Гаусса, умноженной на степенную функцию

М. А. Галыгина

Студент магистрант

Е. А. Киселев

Доцент

Введение

Многие алгоритмы цифровой обработки сигналов, активно используемые в настоящее время, основаны на разложении в ряд по какой-либо системе функций. Это, прежде всего, различные варианты преобразования Фурье [1]. Также в качестве примера можно привести весьма популярную на данный момент теорию вейвлетов [2, 3].

При создании алгоритмов одним из ключевых требований, предъявляемых к системам функций, является наличие удобной структуры. Для работы с цифровыми сигналами достаточно хорошо подходят системы целочисленных сдвигов, поскольку часто сигнал задается на равномерной временной сетке отсчетов. Одной из наиболее востребованных в настоящее время является система целочисленных сдвигов функции Гаусса [4]. В данной работе рассматривается система

сдвигов, порожденная функцией Гаусса, умноженной на степенную. Основной трудностью для построения алгоритмов разложения по данному семейству функций является его неортогональность. Для решения этой проблемы нами построена двойственная система.

1. Постановка задачи и основные соотношения

Рассматривается пространство $L_2(R)$. Скалярное произведение задается следующим образом:

$$(f,g)=\int_{-\infty}^{\infty} f(x)\overline{g(x)}dx, \tag{1}$$

где $\overline{g(x)}$ обозначает комплексное сопряжение функции $g(x)$. Прямое и обратное преобразование Фурье определим следующим образом:

$$\hat{f}(\xi)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(t)e^{-it\xi}dt,\ f(t)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\hat{f}(\xi)e^{it\xi}d\xi. \tag{2}$$

Для простоты все дальнейшие определения и соотношения мы будем формулировать сразу для систем целочисленных сдвигов.

Определение [2, 3]. Два семейства сдвигов $\phi(\cdot-k),\ \ k\in Z$, и $\psi(\cdot-k),\ \ k\in Z$, образуют двойственную систему в пространстве $L_2(R)$, если имеет место равенство

$$(\phi(\cdot-m),\psi(\cdot-n))=\int_{-\infty}^{\infty}\phi(x-m)\overline{\psi(x-n)}dx=\delta_{m,n},m,n\in Z, \tag{3}$$

где $\delta_{m,n}$ – символ Кронекера.

В силу наличия структуры сдвига, условию ортогональности (3) можно придать более краткую форму:

$$\int_{-\infty}^{\infty}\phi(t-k)\overline{\psi(t)}dx=\delta_{k,0},k\in Z. \tag{4}$$

Таким образом, для получения двойственной системы остается только найти функцию $\psi(t)$, удовлетворяющую соотношению (4).

Объектом изучения в данной работе будут системы целочисленных сдвигов, порожденные функцией следующего вида:

$$\phi_n(x)=x^n\exp\left(-\frac{x^2}{2\sigma^2}\right),\sigma>0,n\in N. \tag{5}$$

2. Построение двойственной системы

В случае семейств целочисленных сдвигов существует общая формула для построения двойственной системы [2, 3]:

$$\hat{\psi}(\xi) = \frac{\hat{\phi}(\xi)}{2\pi \sum_{k=-\infty}^{\infty} |\hat{\phi}(\xi + 2\pi k)|^2}. \tag{6}$$

Функция $\psi(t)$, определяемая выражением (6), будет линейной комбинацией $\phi(\cdot - k)$, $k \in Z$. Формула (6) достаточно громоздкая, но это не единственный вариант построения двойственных систем. Если не требовать, чтобы $\psi(t)$ принадлежала линейной оболочке $\phi(\cdot - k)$, $k \in Z$, то существуют и другие функции $\psi(t)$, удовлетворяющие условию (4).

Для краткой записи дальнейших соотношений нам потребуются *B*-сплайны. Рекуррентно они задаются следующим образом [3]:

$$B_1(x) = \chi_{[-0.5,0.5]}(x), \tag{7}$$

$$B_{n+1}(x) = \int_{-\infty}^{\infty} B_n(x-t) B_1(t) dt, n = 1,2,\ldots \ , \tag{8}$$

где $\chi_{[a,b]}(x)$ – характеристическая функция отрезка $[a,b]$. Производную *m*-го порядка по внешнему аргументу функции $B_n(x)$ мы будем кратко обозначать $B_n^{(m)}(x)$.

Теорема. Пусть функция $\psi_n(t)$ в образах Фурье задается выражением

$$\hat{\psi}_n(\xi) = \frac{i^n}{(2\pi)^{n+1} \sigma^{2n+1} n!} \exp\left(\frac{\sigma^2 \xi^2}{2}\right) B_{n+1}^{(n)}\left(\frac{\xi}{2\pi}\right). \tag{9}$$

Тогда имеет место следующее соотношение:

$$\int_{-\infty}^{\infty} \phi_m(x-k) \overline{\psi_n(x-l)} dx = \delta_{k,l} \delta_{m,n}, k,l \in Z, m \le n. \tag{10}$$

Доказательство. Пользуясь формулой (4), перепишем равенство (10) в следующей эквивалентной форме:

$$\int_{-\infty}^{\infty} \phi_m(x-k) \overline{\psi_n(x)} dx = \delta_{k,0} \delta_{m,n}, k, \in Z, m \le n. \tag{11}$$

Затем убедимся в справедливости (11) прямой подстановкой. Для этого рассмотрим сначала следующий вспомогательный интеграл:

$$\lambda_n(y) = \int_{-\infty}^{\infty} \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right) \overline{\psi_n(x)}dx. \quad (12)$$

Воспользуемся унитарностью и свойством преобразования Фурье сдвига:

$$\lambda_n(y) = \int_{-\infty}^{\infty} \sigma \exp\left(-\frac{\sigma^2\xi^2}{2}\right) e^{-iy\xi} \overline{\hat{\psi}_n(\xi)}d\xi. \quad (13)$$

Затем подставим выражение (9) в (13):

$$\lambda_n(y) = \frac{(-i)^n}{2\pi\sigma^{2n}n!} \int_{-\infty}^{\infty} \frac{d^n}{d\xi^n}\left(B_{n+1}\left(\frac{\xi}{2\pi}\right)\right) e^{-iy\xi}d\xi. \quad (14)$$

Применяя далее свойства преобразования Фурье, связанные с производной и масштабированием, а также пользуясь равенством

$$\hat{B}_n(\xi) = \frac{1}{\sqrt{2\pi}}\mathrm{sinc}^n\left(\frac{\xi}{2}\right) = \frac{1}{\sqrt{2\pi}}\frac{\sin^n(\xi/2)}{(\xi/2)^n}, \quad (15)$$

в результате придем к соотношению

$$\lambda_n(y) = \int_{-\infty}^{\infty} \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right) \overline{\psi_n(x)}dx = \frac{1}{\sigma^{2n}n!} y^n \mathrm{sinc}^{n+1}(\pi y). \quad (16)$$

Вычислим производную по y порядка $m \le n$ левой и правой части соотношения (16) в целочисленных точках $y = k$, $k \in Z$. После несложных, но громоздких расчетов придем к равенству

$$\int_{-\infty}^{\infty} \frac{d^m}{dy^m}\left(\exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)\right) \overline{\psi_n(x)}dx = \frac{1}{\sigma^{2n}}\delta_{k,0}\delta_{m,n}, m \le n, y = k \in Z. \quad (17)$$

Далее воспользуемся методом математической индукции. При $m = 0$, согласно формуле (17), сразу получаем требуемое:

$$\int_{-\infty}^{\infty} \exp\left(-\frac{(x-k)^2}{2\sigma^2}\right) \overline{\psi_n(x)}dx = \frac{1}{\sigma^{2n}}\delta_{k,0}\delta_{0,n} = \delta_{k,0}\delta_{0,n}, k \in Z. \quad (18)$$

Предположим, что при всех m, меньших некоторого натурального $m_1 \le n$, соотношение (11) уже доказано, т. е.

$$\int_{-\infty}^{\infty} (x-k)^m \exp\left(-\frac{(x-k)^2}{2\sigma^2}\right) \overline{\psi_n(x)}dx = \delta_{k,0}\delta_{m,n} = 0, k \in Z, m < m_1. \quad (19)$$

Производная под интегралом в формуле (17) представляет из себя функцию Гаусса, умноженную на полином степени m, имеющий своим

аргументом $x-k$. Из (19) следует, что не равный нулю вклад в интеграл при $m=m_1$ может давать только старшая степень $x-k$, т. е. $(x-k)^{m_1}$:

$$\lambda_n^{(m_1)}(k)=\frac{1}{\sigma^{2m_1}}\int_{-\infty}^{\infty}(x-k)^{m_1}\exp\left(-\frac{(x-k)^2}{2\sigma^2}\right)\overline{\psi_n(x)}dx. \qquad (20)$$

С другой стороны, пользуясь формулой (17), получим:

$$\frac{1}{\sigma^{2m_1}}\int_{-\infty}^{\infty}(x-k)^{m_1}\exp\left(-\frac{(x-k)^2}{2\sigma^2}\right)\overline{\psi_n(x)}dx=\frac{1}{\sigma^{2n}}\delta_{k,0}\delta_{m_1,n}. \qquad (21)$$

Отсюда следует требуемое соотношение (11). Теорема доказана.

Согласно теореме, функции $\phi_n(\cdot-k),\ \ k\in Z$, и $\psi_n(\cdot-k),\ \ k\in Z$, образуют двойственную систему. Тогда, если исследуемая функция $f(x)$ принадлежит линейной оболочке $\phi_n(x-k)$

$$f(x)=\sum_{k=-\infty}^{\infty}c_k\phi_n(x-k), \qquad (22)$$

коэффициенты разложения c_k могут быть рассчитаны по формуле

$$c_k=\int_{-\infty}^{\infty}f(x)\overline{\psi_n(x-k)}dx. \qquad (23)$$

Отметим, что построенный математический аппарат позволяет проводить разложение и в более общем случае – если $f(x)$ является суммой функций $\phi_n(x-k)$ с разными значениями $n=0,1,...,N$. Для этого по формуле (23) с помощью $\psi_n(x)$, $n=N$, нужно сначала найти коэффициенты c_N и вычесть из $f(x)$ слагаемые, содержащие сдвиги $\phi_N(x-k)$. Затем необходимо повторить процедуру для $n=N-1$, и т. д.

Заключение

В данной работе представлено решение задачи о разложении по системе сдвигов, порожденной функцией Гаусса, умноженной на степенную. В основе лежит построение двойственной системы, которая в образах Фурье определяется формулой (9). К сожалению, обратное преобразование Фурье найти не удается. Тем не менее, полученная формула вполне пригодна для практического применения: для этого необходимо вычислить преобразование Фурье исследуемого сигнала и воспользоваться свойством унитарности.

В дальнейшем планируется осуществить построение других вариантов двойственных систем, а также протестировать разработанный математический аппарат в расчетах с реальными сигналами.

Список литературы

1. Власова, Е. А. Ряды / Е. А. Власова. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2002. – 611 с.
2. Новиков, И. Я. Теория всплесков / И. Я. Новиков, В. Ю. Протасов, М. А. Скопина. – М. : ФИЗМАТЛИТ, 2005. – 616 с.
3. Чуи, Ч. Введение в вейвлеты / Ч. Чуи; пер. с англ. Я. М. Жилейкина. – М. : Мир, 2001. – 412 с.
4. Maz'ya, V. Approximate Approximations / V. Maz'ya, G. Schmidt. – New York : AMS Mathematical Surveys and Monographs, 2007. – Vol. 141. – 350 p.

Интерполяция с помощью дискретных подсистем когерентных состояний

А. С. Глущенко

Студент магистрант

Е. А. Киселев

Доцент

Введение

В настоящее время значительное внимание уделяется различным вариантам оконного преобразования Фурье [1]. При этом большинство разработанных на данный момент методов в этой области основано на использовании каких-либо интегральных соотношений [2, 3]. Однако для работы с цифровыми сигналами более удобным представляется интерполяционный подход, аналогичный дискретному преобразованию Фурье (ДПФ). Созданию соответствующего математического аппарата посвящена данная работа.

Когерентными состояниями гармонического осциллятора в квантовой физике называются функции вида

$$g(x) = \exp\left(-\frac{(x-a)^2}{2\sigma^2}\right)e^{ibx}, \qquad (1)$$

где a, b – вещественные числа, $\sigma > 0$ [4]. В данной работе мы будем строить интерполяционные формулы на основе двухпараметрического семейства функций

$$g_{p,r}(x) = \exp\left(-\frac{(x-\alpha_1 p)^2}{2\sigma^2}\right) e^{i\alpha_2 rx}, p, r \in Z, \quad (2)$$

где α_1, α_2 – некоторые фиксированные положительные вещественные параметры. При условии $\alpha_1\alpha_2 < 2\pi$ система функций (2) переполнена в $L_2(R)$ и образует так называемый фрейм Габора [1].

Выберем узловые точки интерполяции целочисленными: $x_k = k$, $k \in Z$, поскольку измерения на практике часто производят с постоянным шагом, который, не ограничивая общности, можно принять равным единице. Требуется построить линейную комбинацию $f(x)$ функций $g_{p,r}(x)$, которая в узлах x_k будет совпадать с некоторой исследуемой функцией $f(x)$. Для решения задачи интерполяции весьма удобным инструментом являются узловые функции.

Определение. Функция $\phi_k(x)$, являющаяся линейной комбинацией $g_{p,r}(x)$, называется узловой для точки x_k, если она удовлетворяет соотношению

$$\phi_k(x_m) = \delta_{k,m}, m \in Z, \quad (3)$$

где $\delta_{k,m}$ – символ Кронекера.

Если в нашем распоряжении имеются узловые функции для каждой точки x_k, то интерполирующая функция $f(x)$ легко строится по формуле

$$f(x) = \sum_{k=-\infty}^{\infty} f(x_k)\phi_k(x). \quad (4)$$

1. Построение узловых функций

Известен следующий результат [5]: функция

$$\phi(x,\sigma) = \sum_{p=-\infty}^{\infty} c_p(\sigma)\exp\left(-\frac{(x-p)^2}{2\sigma^2}\right), \sigma > 0, \quad (5)$$

где коэффициенты $c_p(\sigma)$ определяются выражениями

$$c_p(\sigma) = \frac{1}{K(\sigma)}\exp\left(\frac{p^2}{2\sigma^2}\right)\sum_{r=|p|}^{\infty}(-1)^r \exp\left(-\frac{(r+0.5)^2}{2\sigma^2}\right), \quad (6)$$

$$K(\sigma) = \sum_{r=-\infty}^{\infty}(4r+1)\exp\left(-\frac{(2r+0.5)^2}{2\sigma^2}\right), \quad (7)$$

в целочисленных точках принимает следующие значения:

$$\%(k,\sigma) = \delta_{k,0}, k \in Z. \quad (8)$$

Приведенное утверждение означает, что функция $\%(x,\sigma)$ является узловой для точки $x_0 = 0$ в случае $\alpha_1 = 1$. Все остальные узловые функции можно тогда получить из $\%(x,\sigma)$ операцией сдвига: $\%_k(x) = \%(x-k,\sigma)$. Это приводит к хорошо изученной задаче об интерполяции с помощью системы целочисленных сдвигов функции Гаусса [5, 6].

Ключевым преимуществом данного подхода является его простота, но он обладает и рядом недостатков. Во-первых, на практике при $\sigma > 2$ появляется значительная вычислительная неустойчивость [6]. Во-вторых, при разложении задействованными окажутся далеко не все функции из набора (2). Между тем, использование не только сдвигов, но и частотных компонент способно обеспечить более гибкий математический аппарат для анализа цифровых сигналов.

Для решения описанных выше проблем мы предлагаем построить несколько иное семейство узловых функций. Зададим натуральное число n и рассмотрим функцию

$$\lambda_n(x,\sigma) = \%\left(\frac{x}{n},\frac{\sigma}{n}\right) = \sum_{p=-\infty}^{\infty} c_p\left(\frac{\sigma}{n}\right)\exp\left(-\frac{(x-np)^2}{2\sigma^2}\right). \quad (9)$$

При надлежащем выборе n параметр σ/n в аргументе $c_p(\sigma/n)$ может быть сделан меньшим 2. Функция (9) при этом уже не будет удовлетворять равенству (8) во всех целых точках, вместо него имеем

$$\lambda_n(nm,\sigma) = \delta_{m,0}, m \in Z. \quad (10)$$

Чтобы вновь добиться выполнения условия (8), умножим $\lambda_n(x,\sigma)$ на тригонометрический полином

$$\frac{\sin(\pi x)}{n\sin(\pi x/n)} = \frac{1}{n}\sum_{k=0}^{n-1}\exp\left(\frac{i\pi(-(n-1)+2k)x}{n}\right). \quad (11)$$

Тогда, как легко убедиться, полученная функция

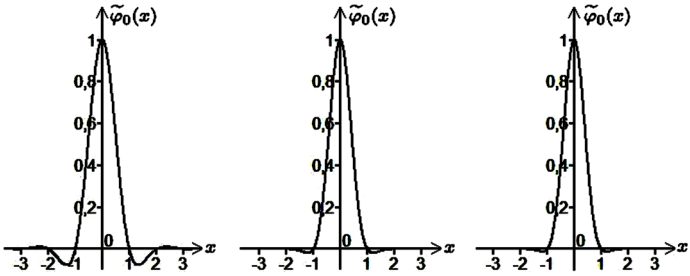
$$\%_0(x) = \lambda_n(x,\sigma)\frac{\sin(\pi x)}{n\sin(\pi x/n)} \quad (12)$$

будет являться узловой для точки $x_0 = 0$, т. е. соотношение (8) для нее выполнено. При этом $\%_0(x)$ является линейной комбинацией функций, принадлежащих семейству (2) с параметрами $\alpha_1 = 1$, $\alpha_2 = \pi/n$.

Остальные узловые функции, как и ранее, можно построить, применив к $\tilde{\varphi}_0(x)$ операцию целочисленного сдвига.

2. Результаты расчетов

Приведем несколько примеров графиков узловой функции (12) при различных значениях параметров σ и n. На рис. 1 представлен случай $\sigma = 0.5$. Можно отметить, что при увеличении n функция становится лучше локализованной по переменной x.



а) б) в)

Рис. 1. Функция $\tilde{\varphi}_0(x)$ при $\sigma = 0.5$:
а – $n = 1$, б – $n = 2$, в – $n = 3$

Далее, на рис. 2 показан случай $\sigma = 1.5$. В целом наблюдаются аналогичные эффекты. Локализация функций по x несколько хуже.

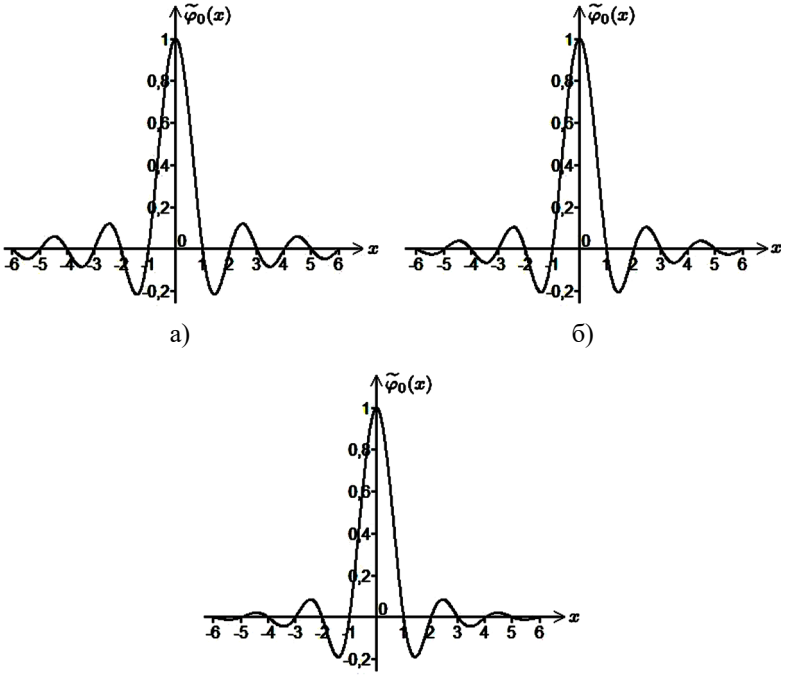
Заключение

В данной работе предложен новый способ представления функций в виде ряда по дискретным подсистемам когерентных состояний, заданных на прямоугольной решетке. В основе него лежит интерполяционный подход. Его преимуществом является то, что он не требует вычисления интегралов, которые на практике обычно приходится находить приближенно по квадратурным формулам.

Для реализации процедуры интерполяции было построено новое семейство узловых функций, которое имеет достаточно удобную для практики структуру сдвигов. Отметим, что рассмотренные в данной работе наборы узловых функций определяются не единственным образом. Например, некоторые из них могут быть дополнительно умножены на $e^{i\alpha_2 rx}$, $r \in Z$, что не нарушит условие (3).

В дальнейшем планируется рассмотреть другие варианты реализации интерполяционного подхода, исследовать его

вычислительные особенности в зависимости от параметров σ и n, а также применить разработанный математический аппарат для анализа и обработки цифровых сигналов.



а) б)

в)

Рис. 2. Функция $\%_0(x)$ при $\sigma = 1.5$:
а – $n = 1$, б – $n = 2$, в – $n = 3$

Список литературы

1. Feichtinger, H. G. A Guided Tour from Linear Algebra to the Foundations of Gabor Analysis / H. G. Feichtinger, F. Luef, T. Werther // Gabor and Wavelet Frames. Lecture Notes Series Institute for Mathematical Sciences (NUS). – 2007. – Vol. 10. – P. 1-49.
2. Janssen, A. J. E. M. Duality and Biorthogonality for Weyl-Heisenberg Frames / A. J. E. M. Janssen // Journal of Fourier Analysis and Applications 1. – 1995. – № 4. – P. 403-436.
3. Минин, Л. А. О разложении по фреймам Габора, порожденным функцией Гаусса / Л. А. Минин, И. Я. Новиков,

С. Н. Ушаков // Математические заметки. – 2016. – Т. 100. – № 6. – С. 890-892.

4. Глаубер, Р. Оптическая когерентность и статистика фотонов : курс лекций / Р. Глаубер. – М. : Мир, 1996. – 189 с.

5. Maz'ya, V. Approximate Approximations / V. Maz'ya, G. Schmidt // AMS Mathematical Surveys and Monographs. – New York, 2007. – Vol. 141. – 350 p.

6. Журавлев, М. В. О вычислительных особенностях интерполяции с помощью систем целочисленных сдвигов гауссовых функций / М. В. Журавлев, Л. А. Минин, С. М. Ситник // Научные ведомости Белгородского государственного университета. – 2009. – № 13(68), вып. 17/2. – С. 89-99.

Концепция и архитектура адаптивного семантического органайзера

И. С. Господарикова

Студент магистрант

В. С. Господарикова

Студент бакалавриата

С. А. Никонова

Студент бакалавриата

В. В. Гаршина

Доцент

Введение

В настоящей статье речь идёт о предлагаемой нами концепции адаптивного семантического органайзера, о её практическом значении и о задачах, которые возникают при попытке создания подобного программного продукта.

Проанализировав некоторые современные голосовые ассистенты [1], мы пришли к выводу, что одним из возможных направлений развития подобных программных продуктов можно считать создание персонифицированной системы знаний о пользователе, которая бы способствовала повышению качества взаимодействия с

голосовым ассистентом. Помощник, снабжённый такой системой знаний, более успешно, на наш взгляд, адаптировался бы к пожеланиям пользователя, поскольку хранимая там информация может успешно применяться для подбора интересных новостей, поиска объявлений, напоминаний о делах и встречах и т.п.

Данную идею мы взяли за направление деятельности, и теперь нашу задачу в рамках настоящей статьи мы видим в том, чтобы сформулировать основные идеи предлагаемой концепции, обозначить проблемы, которые могут возникнуть при построении такого голосового помощника, а также представить некоторые результаты нашей работы по указанному направлению.

1. Концепция адаптивного семантического органайзера

Как говорилось во введении, одной из целей статьи является освещение концепции адаптивного семантического органайзера. Для наглядности мы изложили свои представления о данном проекте в виде схемы, которую вы можете увидеть на рис. 1.

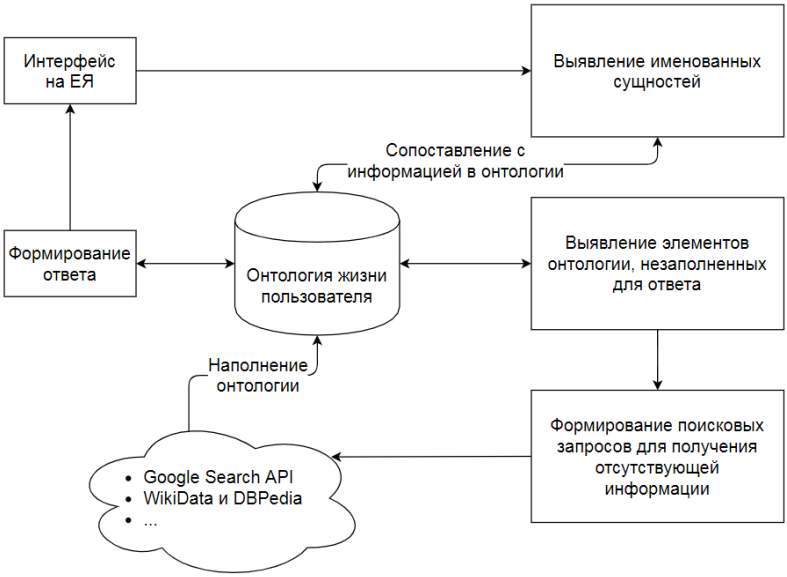


Рис. 1. Схема работы адаптивного семантического органайзера

Из данной схемы можно понять, что центральным элементом адаптивного семантического органайзера является онтология, содержащая данные о пользователе. Она условно названа нами

онтологией жизни пользователя. В соответствии с хранящимися в ней фактами происходит работа описываемого программного продукта.

Получаемая от пользователя через голосовой интерфейс информация преобразуется в текст, текст этот проходит этапы анализа, которые могут быть реализованы, например, посредством Томита-парсер [2]. В том числе следует проводить семантический анализ, то есть выявление сущностей и связей между ними. На основании вычлененного таким образом смысла высказывания, происходит сопоставление его с уже известными органайзеру фактами. По итогу может произойти выявление нового факта, формирование ответной фразы на смысловом уровне или же обнаружение факта, недостающего в онтологии для ответа. В первом случае факт, если он не противоречит данным в онтологии, может быть добавлен в неё. При противоречии следует разрешать вопрос о том, насколько противоречащий факт из онтологии актуален. Имеется в виду возможность изменения фактов по ходу работы и перезапись данных в онтологии. Второй случай – формирование ответа – влечёт за собой генерацию уже конкретной фразы и озвучение её пользователю. Третий вариант сопряжён с выявлением недостающего факта, и, по нашим представлениям, должен запускать сценарий добывания информации из источников в сети Интернет. В частности, нам видится реализация данного сценария через формирование запросов к Google Search API [3] или же к таким ресурсам как Wikidata [4] или DBpedia [5]. Таким образом, предполагаемая система будет иметь инструмент для получения недостающих знаний, и пользователю не придётся задавать какие-то простейшие факты самому.

2. Базовая онтология жизни

В нашем видении концепции адаптивного семантического органайзера, онтология жизни занимает центральное место. Конечно, не меньшее значение для итогового качества имеет вопрос распознавания звучащей речи и преобразование её в текст, однако основную новизну проекта представляет наполняемая система знаний, которую мы и будем рассматривать в первую очередь.

Многие аспекты в жизнях самых разных, казалось бы, людей в наше время совпадают. То есть на определённом уровне абстракции все пользователи будут заинтересованы в хранении информации одинакового качества. Мы провели исследования с целью создать хотя бы примерный макет жизни современного человека. Для этого жизнь человека была поделена нами на отдельные сферы: дом и быт, работа, отдых (досуг), друзья и семья, здоровье – и воплощена в примерной онтологии этих сфер [6, 7], составленной при помощи программного

средства Protégé [8]. На рис. 2, представленном ниже, можно увидеть онтограф получившейся модели.

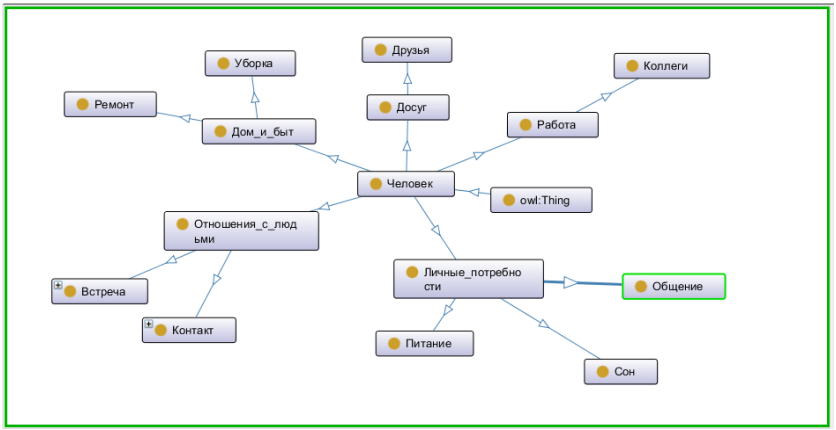


Рис. 2. Упрощённый пример онтологии жизни человека

В данной онтологии присутствует класс «Человек». Объекты этого класса могут обозначать как человека, которому принадлежит органайзер, так и других людей с которым он связан какими-либо отношениями. Таких людей, информацию о них и встречи с ними мы вынесли в сферу, которую условно обозначили «Друзья и семья». Вы можете видеть отношения этих и других классов на онтографе.

В свою очередь, «Дом и быт» включает в себя все аспекты, связанные с домом человека. То есть его жилищем, уборкой в нём, продуктовыми запасами и многим другим. «Работа» подразумевает под собой профессиональную деятельность человека, которая приносит ему доход. Также эта сфера включает рабочие отношения и встречи, покупку необходимых канцтоваров или тому подобных атрибутов. Следующая сфера – это «Досуг». Сюда входят хобби, встречи с друзьями и семьей на выходных, походы в кино и прочие увлечения. Последней ещё не рассмотренной нами сферой является «Здоровье» или личные потребности человека. Сюда включены все действия, совершаемые человеком для сохранения своего психического, физиологического и духовного здоровья человека. Это, например, сон, питание, гигиена, общение и прочее.

Таким образом, мы составили онтологию, примерно описывающую жизнь человека и включающую в себя пять сфер, которые скрывают под собой множество разделов, состоящих из множества процессов и

действий. Однако в то же время можно заметить некоторое несовершенство такой модели. Если мы попытаемся записать в неё, например, поход в магазин с подругой, то можем затронуть сразу, три сущности: «Отношения с людьми» (с подругой ещё нужно созвониться и договориться о встрече), «Досуг», а также «Здоровье» (как потребность в общении). Если же мы покупаем в магазине что-либо для дома или работы, то затрагиваем сразу 4-5 сфер. Безусловно, в таком рассмотрении данная онтология уже кажется некорректной и довольно узкой. Но она была создана лишь для примера. Её можно начать расширять, и на одном из этапов дойти до более совершенной версии, например, такой как представлена на рис. 3.

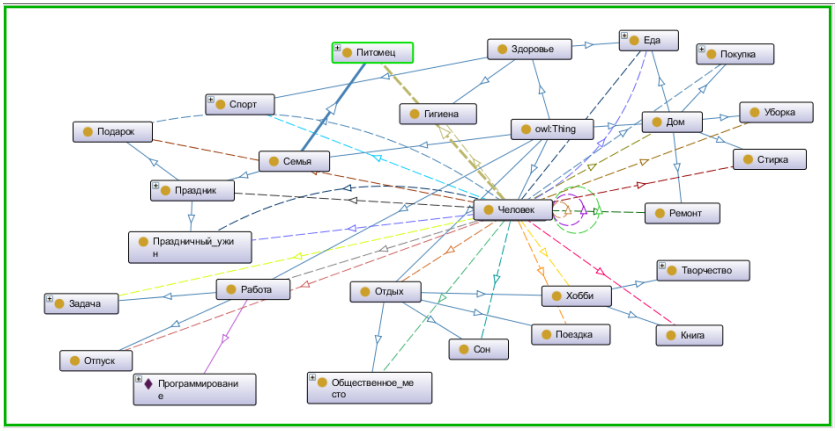


Рис. 3. Более сложный пример онтологии жизни

По ней видно, что дальнейшее усложнение модели сильно снижает читабельность онтографа, так как, помимо огромного числа сущностей, он имеет в перспективе развития ещё и высокую степень связности. Притом заметим, что данная онтология всё ещё не очень успешно отражает все аспекты жизни, и, например, указания того, что сущности «Питомец» нужна сущность «Еда» в модели не представлено.

Тем не менее, если понять принцип построения самой модели и вдумчиво определить основные сущности, то создать её вполне реально. Тогда информацию, вычлененную из речи пользователя, можно будет запомнить хотя бы на уровне фактов, и в соответствии с ними давать некоторый взвешенный и обдуманный ответ при взаимодействии с пользователем.

Заключение

В рамках данной статьи была изложена концепция адаптивного семантического органайзера, а кроме того показана актуальность данной темы для исследований. Были приведены результаты моделирования онтологии жизни человека. По ним мы можем сказать, что процесс создания базовой онтологии жизни в полном её объёме довольно трудоёмок, хотя препятствия для этого не являются непреодолимыми. Зато такой подход к разработке голосового ассистента делает возможным создание программного продукта, ориентированного на конкретного пользователя и способного генерировать действительно полезную информацию с опорой на ранее данные факты, а не просто быть средством для развлечения.

Список литературы

1. Господарикова И. С. Существующие технологии анализа речи на естественном языке / И. С. Господарикова, В. В. Гаршина // Информатика: проблемы, методы, технологии : материалы XX Международной научно-методической конференции (Воронеж, 13-14 февраля 2020 г.). – Воронеж, 2020. – С. 1586-1590.
2. Томита-парсер [Электронный ресурс] : документация. – Режим доступа : https://yandex.ru/dev/tomita/
3. Google Search API [Электронный ресурс] : документация. – Режим доступа : https://rapidapi.com/apigeek/api/google-search3
4. Wikidata: Доступ к данным [Электронный ресурс] : информационная страница. – Режим доступа : https://www.wikidata.org/wiki/Wikidata:Data_access/ru
5. DBpedia [Электронный ресурс] : база данных. – Режим доступа : https://wiki.dbpedia.org/
6. Стандарт OWL [Электронный ресурс] : информационная страница. – Режим доступа : https://www.w3.org/OWL/
7. Онтологическое моделирование предприятий: методы и технологии : монография / С. В. Горшков [и др.] ; отв. ред. С. В. Горшков. – Екатеринбург : Изд-во Урал. ун-та, 2019. – 236 с.
8. Protégé [Электронный ресурс] : программное обеспечение. – Режим доступа : https://protege.stanford.edu/

Разработка нейросетевого классификатора эмоций

Е. А. Гребенщикова

Студент магистрант

Е. Ю. Митрофанова

Доцент

Введение

Выражения лица являются жизненно важными идентификаторами человеческих чувств, потому что они соответствуют эмоциям. В большинстве случаев (примерно в 55% случаев) [1] выражение лица является невербальным способом выражения эмоций, и его можно рассматривать как конкретное доказательство того, что человек говорит правду или нет [2]. Текущие подходы в основном сосредоточены на лицевом расследовании, сохраняя фон нетронутыми, и, следовательно, создали множество ненужных и вводящих в заблуждение особенностей, которые путают процесс обучения CNN. В данной статье рассмотрены пять основных описанных классах выражения лица: недовольство / гнев, грусть / несчастье, улыбка / радость, страх и удивление [3]. Алгоритм FERC предназначен для экспрессивного изучения и для характеристики данного изображения на эти пять основных классов эмоций. Отмеченные методы обнаружения выражений лица могут быть описаны как два основных подхода. Первое – это отличительные выражения [4], которые отождествляются с явным классификатором, а второе – делает характеристику зависимой от выделенных элементов лица [5]. В системе кодирования действий лица (FACS) [6] единицы действия используются в качестве маркеров выражения. Эти маркеры были различимы по изменениям мышц лица.

1. Методология

Сверточная нейронная сеть (CNN) является наиболее популярным способом анализа изображений. CNN отличается от многослойного персептрона (MLP) тем, что они имеют скрытые слои, называемые сверточными слоями. Предложенный метод основан на двухуровневой структуре CNN. Первый рекомендуемый уровень – это удаление фона [7], используемое для извлечения эмоций из изображения, как показано на рис. 1. Здесь обычный сетевой модуль CNN используется для извлечения первичного вектора экспрессии (EV). Вектор экспрессии (EV) генерируется путем отслеживания значимых точек лица. EV

напрямую связан с изменениями в выражении. EV получается с использованием базового блока персептрона, нанесенного на фоновое изображение лица на рис. 2. В предложенной модели FERC мы также имеем некорпоративный слой персептрона в качестве последней стадии. Каждый из сверточных слоев получает входные данные (или изображение), преобразует их и затем выводит их на следующий уровень. Это преобразование является операцией свертки, как показано на рис. 3. Все используемые сверточные слои способны обнаруживать структуры. В каждом сверточном слое использовались четыре фильтра. Входное изображение, подаваемое в первую часть CNN (используется для удаления фона), обычно состоит из фигур, ребер, текстур и объектов вместе с гранью. В начале сверточного слоя 1 используются краевые детекторы, детекторы кругов и детекторы углов. Как только лицо обнаружено, второй CNN-фильтр обнаруживает такие черты лица, как глаза, уши, губы, нос и щеки. Фильтры обнаружения края, используемые в этом слое, показаны на рис. 4. CNN второй части состоит из слоев с матрицей ядра 3 × 3, например, [0.25, 0.17, 0.9; 0.89, 0.36. 0,63; 0.7, 0.24, 0.82]. Эти числа изначально выбираются между 0 и 1. Как только фильтр настроен с помощью контролирующего обучения, он применяется к удаленному фону лицу (т.е. к выходному изображению CNN первой части) для обнаружения различных частей лица (например, глаз, губ, носа, ушей и так далее).

2. Удаление фона

После получения входного изображения применяется алгоритм определения тона кожи [8] для извлечения частей тела человека из изображения. Это выходное изображение, детектированное по тону кожи, является двоичным изображением и используется в качестве признака для первого слоя удаления фона CNN (также упоминаемого как CNN первой части в этой рукописи).

На рис. 1 взято входное изображение. Затем входное изображение передается в первую часть CNN для удаления фона. После удаления фона генерируется экспрессионный вектор лица EV на рис. 5. Другой CNN (CNN второй части) применяется с надзорной моделью, полученной из базы данных «правда-земля». Наконец, эмоция от текущего входного изображения обнаружена.

На рис. 2 нос (N), губа (P), лоб (F), глаза (Y) отмечены с использованием обнаружения краев и сопоставления ближайших скоплений. Положение влево, вправо и центр представлены с использованием L, R и C, соответственно.

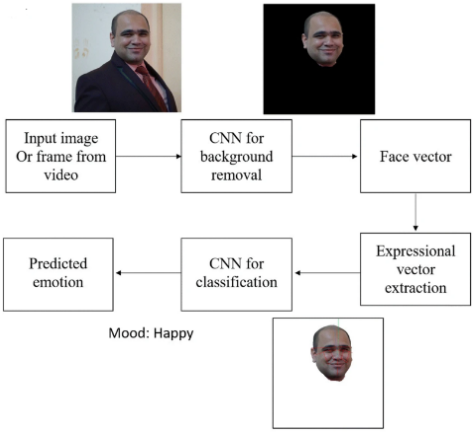


Рис. 1. Структурная схема FERC

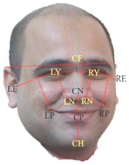


Рис. 2. Векторы лица отмечены на удаленном фоне лица

На рис. 3 изображена работа фильтра свертки с ядром 3 × 3. Каждый пиксель из входного изображения и его восемь соседних пикселей умножаются на соответствующее значение в матрице ядра, и, наконец, все умноженные значения складываются вместе для достижения окончательного выходного значения.

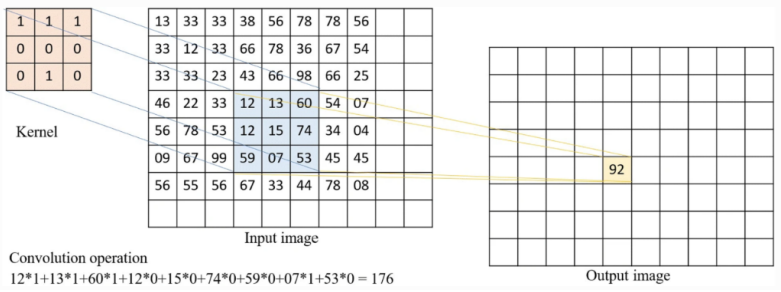


Рис. 3. Сверточный фильтр

Это определение тона кожи зависит от типа входного изображения. Если изображение является цветным изображением, то можно использовать порог цвета YCbCr. Для кожного покрова значение Y должно быть больше 80, Cb должно находиться в диапазоне от 85 до 140, значение Cr должно быть в диапазоне от 135 до 200.

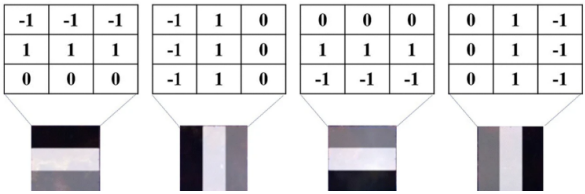


Рис. 4. Матрица фильтра детектора краев

22	7	7	8	4	4	9	9	7	7	2	9	8	6	6	4	5	5	4	11	12	4	10	6
LE-RE	LE-LY	RE-RY	RY-LY	RY-CN	LY-CN	LE-CN	RE-CN	LE-LP	RE-RP	CN-CP	CP-LE	CP-RE	CF-RE	CF-LE	CF-CN	CF-LY	CF-RY	LH-LE	LH-RE	RH-LE	RH-CE	LH-RH	CF-CP

L: Left RE: Right C: Center Y: Eye E: Ear N: Nose P: Lip F: forehead H:Cheek

Рис. 5. Пример матрицы EV

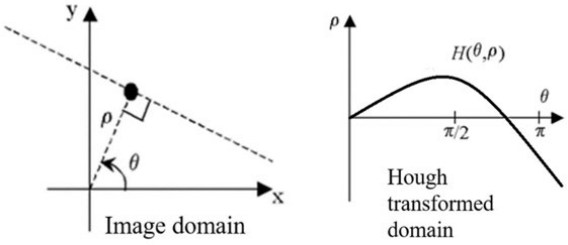


Рис. 6. Представление точки в области изображений

Набор значений, упомянутых в приведенной выше строке, был выбран методом проб и ошибок и работал практически для всех доступных оттенков кожи. Было обнаружено, что если входное изображение имеет оттенки серого, то алгоритм определения тона кожи имеет очень низкую точность. Чтобы повысить точность при удалении фона, CNN также использует фильтр кругов в круге. Эта операция фильтра использует значения преобразования Хафа для каждого

обнаружения круга. Чтобы сохранить однородность независимо от типа входного изображения, преобразование Хафа (рис. 6) всегда использовалось в качестве второй входной функции для удаления фона CNN.

3. Сверточный фильтр

Как показано на рис. 3 для каждой операции свертки, все изображение делится на перекрывающиеся матрицы 3 × 3, а затем соответствующий фильтр 3 × 3 сворачивается по каждой матрице 3 × 3, полученной из изображения. Операция скольжения и взятия точечного произведения называется «сверткой» и, следовательно, называется «сверточным фильтром». Во время свертки точечное произведение обеих матриц 3 × 3 вычисляется и сохраняется в соответствующем месте, например, (1,1) на выходе, как показано на рис. 3. Как только вся выходная матрица рассчитана, этот выходной сигнал передается на следующий уровень CNN для следующего цикла свертки. Последним слоем черты лица, извлекающей CNN, является простой персептрон, который пытается оптимизировать значения масштабного коэффициента и показателя степени в зависимости от отклонения от истинности оснований.

4. Результаты

Для анализа производительности алгоритма использовался собственные набор изображений лиц людей. Было сохранено 70% изображений набора данных 10000 в качестве обучающих изображений и 30% изображений набора данных в качестве тестовых изображений. Всего было выполнено 25 итераций, каждый раз с разными наборами 70% обучающих данных. Наконец, ошибка была рассчитана как стандартное отклонение. На рис. 7 показана оптимизация количества слоев для CNN. Для простоты было сохранено количество слоев и количество фильтров для удаления фона CNN (CNN первой части), а также извлечения элементов лица CNN (CNN второй части) одинаковыми. В этом исследовании варьировали количество слоев от 1 до 8. Было обнаружено, что максимальная точность была около 4. На рисунке 8 показано количество фильтров оптимизации для обоих слоев. Опять же, 1-8 фильтров были опробованы для каждой из четырехслойных сетей CNN. Было обнаружено, что четыре фильтра дают хорошую точность (рис. 8). Следовательно, FERC был разработан с четырьмя слоями и четырьмя фильтрами. На рис. 9 и е показаны обычные фронтальные случаи с сердитыми и неожиданными эмоциями, и алгоритм мог легко их обнаружить (рис. 9). Изображение с поворотом головы, как показано на рис. 9, было сложным из-за ориентации. К

счастью, с вектором пространственных объектов EV с 24 измерениями можно правильно классифицировать ориентированные на 30° грани, используя FERC. У этого метода есть некоторые ограничения, такие как высокая вычислительная мощность во время настройки CNN, а также проблемы с волосами на лице. Но, кроме этих проблем, точность алгоритма очень высока – 96%.

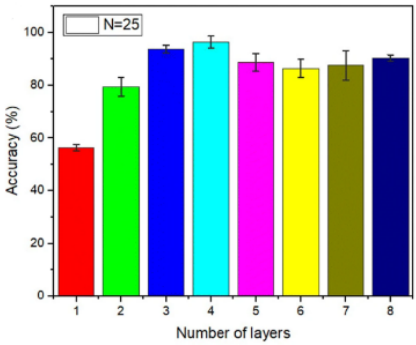


Рис. 7. Оптимизация по количеству слоев CNN (максимальная точность была достигнута для четырехслойного CNN)

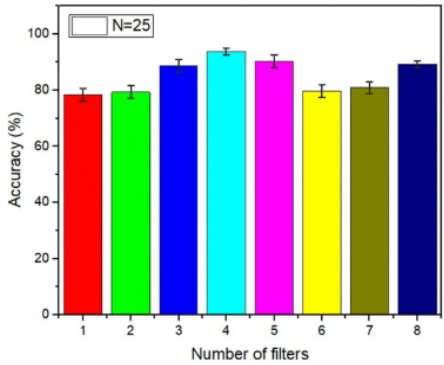


Рис. 8. Оптимизация по количеству фильтров

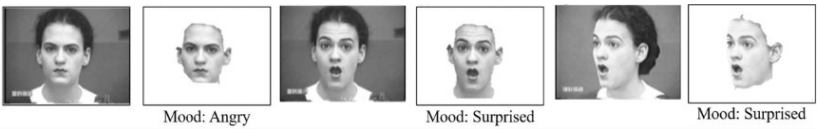


Рис. 9. Вывод удаления фона с прогнозируемой эмоцией

Заключение

FERC – это новый способ обнаружения эмоций на лице, который использует преимущества CNN и контролируемого обучения (возможно благодаря большим объемам данных). Основное преимущество алгоритма FERC заключается в том, что он работает с разными ориентациями (менее 30°) благодаря уникальной 24-значной матрице характеристик EV. Удаление фона добавило большое преимущество в точном определении эмоций. FERC может быть начальным шагом для многих приложений, основанных на эмоциях, таких как детектор лжи, а также обучение на основе настроения для студентов и т. д.

Список литературы

1. Меграбян, А. Невербальная коммуникация / А. Меграбян. – М. : Речь, 2001. – 256 с.
2. Бартлетт, М. Интеллектуальный анализ лица с автоматическим кодированием выражений / М. Бартлетт. – М. : Springer, 2008. – 261 с.
3. Рассел, Дж. Есть ли всеобщее признание эмоций по выражению лица? / Дж. Рассел // Psychol Bull. – 1994. – №1. – С. 102-109.
4. Гизатдинова, Ю. Автоматическое обнаружение лицевых ориентиров по AU / Ю. Гизатдинова // IEEE Transactions on Image Processing. – 2007. – №1. – С. 419-424.
5. Лю, Ю. Распознавание выражений лица с помощью слияния / Ю. Лю // Sensors. – 2017. – №4. – С. 712-718.
6. Экман, Р. Что показывает лицо: фундаментальные и прикладные исследования (FACS) / Р. Экман. – М. : Издательство Оксфордского университета, 1997. – 185 с.
7. Кови, Р. Распознавание эмоций в человеко-компьютерном взаимодействии / Р. Кови // IEEE Signal Processing Magazine. – 2001. – №1. – С. 32-37.
8. Хсу, Р. Распознавание лиц на цветных изображениях / Р. Хсу // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2002. – №5. – С. 696-703.

Анализ и модификация алгоритмов обнаружения текстуально совпадающей информации

М. А. Грибкова

Студент магистрант

В. А. Степанцов

Доцент

Введение

В настоящее время мы живем в мире цифровой информации и постоянно сталкиваемся с вопросами, связанными с ее обменом и передачей. В то же время очень одним из наиболее важных вопросов является защита информации.

Проблемы информационной безопасности рассматриваются и изучаются с разных сторон, существует множество работ, изучающих данную тему, однако, большинство из них посвящено изучению преднамеренных случаев нарушений.

Но, как ни странно, значительную долю случаев угроз безопасности составляют именно непреднамеренные действия, например, сотрудников, не подразумевающие целенаправленное желание нарушить целостность и защищенность данных, с которыми им приходится работать.

Таким образом, одной из серьезных проблем являются вопросы защиты конфиденциальной информации от случайных утечек. Данная ситуация также усложняется тем, что не существует четкого определения, формулировок или ограничений, какая именно информация считается конфиденциальной. На законодательном уровне зафиксировано множество категорий данных, не подлежащих к распространению, имеющих разную структуру и представление.Все это усложняет задачу определения не подлежащей к разглашению информации в, например, пересылаемых документах, что может вести к нарушению ее конфиденциальности.

Данная работа посвящена анализу алгоритмов обнаружения текстуально совпадающей информации и их модификации для возможности использования в рамках узконаправленной задачи.

1. Анализ алгоритмов

Стандартный алгоритм. Данный алгоритм является самым простейшим, но и, зачастую, наиболее трудозатратным. Проверка осуществляется сравнением первого символа исходного текста с первым символом шаблонной подстроки. Если обнаружено совпадение, то происходит переход ко второму символу текста и второму символу подстроки. При совпадении выравниваются следующие символы. Процесс продолжается, пока не будет найден несовпадающий символ, или же пока не пока не будет завершена проверка всей подстроки, то есть не будет найдено вхождение подстроки в исходный текст. В случае второго варианта мы считаем, что задача решена. В случае же первого варианта осуществляется сдвиг указателя в исходном тексте, и процесс поиска подстроки начинается заново.

В худшем случае при наличии строки *S*длиной *m* и подстроки длиной n сложность алгоритма можно записать как *O(n*m)*. Кроме того, к недостаткам можно отнести неиспользование при сдвиге результатов, полученных в ходе предыдущих сравнений.

Алгоритм Кнута-Морриса-Пратта (КМП). При работе с данным алгоритмом используются индексы: индекс по строке *I* и индекс по подстроке *i*.

В отличие от стандартного алгоритма в случае обнаружения несовпадений, при использовании алгоритма КМП откат индекса по строке никогда не происходит, а осуществляется сдвиг подстроки на определенное количество символов [1].

Работа алгоритма состоит из 2 этапов:

1. Формирование массива π, используемого при сдвиге подстроки.
2. Непосредственно поиск подстроки в строке.

Рассмотрим подробно 1-ый этап. Для создания массива π для каждого *i*-того символа подстроки необходима префикс-функция, значение которой хранится в π[*i*].

Префикс функция от строки *S* и позиции *i* в ней – длина *k* наибольшего собственного (не равного всей подстроке), префикса подстроки *S* [1..*i*], который одновременно является суффиксом этой подстроки. То есть в начале строки *S* [1..*i*] длины *i* необходимо найти такой префикс максимальной длины *k*<*i*, который был бы суффиксом данной подстроки *S* [1..*k*]=*S* [(*i*-*k*+1)..*I*].Префикс функция обозначается как π(*S, i*), где *S* – строка, $1 \le i \le S$ – длина подстроки.

На втором этапе происходит поиск подстроки в строке. При совпадении символов происходит сдвиг обоих индексов вправо. При несовпадении используется подготовленный на 1-ом этапе массив π.

Таким образом, видно, что данный алгоритм позволяет уменьшить количество проходов и сравнений, необходимых для проверки строки, что позволяет иметь стабильную скорость работы на длинных строках. Кроме того, эффективность алгоритма не снижается при обработке «неудачных» данных. При наличии строки S длиной m и подстроки длиной n сложность алгоритма может быть записана как $O(n+m)$. Однако к недостаткам можно отнести то, что при нахождении несовпадений возможно осуществление сдвига только на 1 символ.

Алгоритм Бойера-Мура. Работа данного алгоритма также состоит из двух этапов:

– Формирование таблицы смещения d, используемой при сдвиге подстроки.
– Непосредственно поиск подстроки в строке.

Данный алгоритм отличается тем, что сравнение ведется не с начала подстроки, а с ее конца. Это позволяет пропускать те части строки, которые заведомо не будут участвовать в удачном сопоставлении.

На первом этапе составляется таблица d, которая содержит значения удаленности каждого символа от конца подстроки. Если встречаются повторы – более удаленному символу присваивается значение, равное значению для менее удаленного символа. Последнему символу соответствует значение, равное длине подстроки, если он ни разу не встречался ранее. Всем остальным символам, не встречающимся в подстроки, также присваивается значение, равное длине подстроки.

Данный алгоритм неплохо подходит для обработки длинных текстов, но не оправдывает себя на коротких. При наличии строки S длиной m и подстроки длиной n метод показывает сложность $O(n/m)$ при благоприятных, однако же, дает неплохие результаты даже на «неудачных» текстах имея сложность $O((n-m+1)*m+p)$, где p – мощность алфавита.

Алгоритм жадного строкового замощения. Алгоритм жадного строкового замощения получает на вход две строки символов, а на выходе возвращает набор их общих непересекающихся подстрок. Подстрока, входящая в этот набор, называется тайлом (*Tile*).

Введем несколько определений и рассмотрим строк P и T - представления сравниваемых текстовых файлов.

MinimumMatchLength – минимальная длина наибольшего общего префикса строк *Pp* и *Tt*, при которой он учитывается алгоритмом.

MaxMatch–длина самого большого из пока найденных на текущей итерации алгоритма общих префиксов строк *Pp* и *Tt*.

Tiles– набор тайлов.

Mathes– множество, содержащее кандидаты на попадание в набор тайлов.

Алгоритм может быть разделен на два этапа.

1. Поиск наибольших общих подстрок *P* и *T*, состоящих только из непомеченных элементов (в начале работы алгоритма все элементы не помечены). Для этого используются три вложенных цикла: по всевозможным *Pp*, по всем *Tt*, наибольший общий префикс *Pp* и *Tt*. Происходит проверка отношения *MaxMatch* и найденного префикса:
   - если *MaxMatch* меньше, чем длина ранее найденного префикса, то все найденные до этого совпадения удаляются из списка общих подстрок, и туда добавляется найденный префикс;
   - если *MaxMatch* больше длины найденного, то ничего не происходит;
   - если *MaxMatch* равно длине найденного префикса, то этот префикс добавляется в список *Matches*.
2. В процессе второго этапа выполняется проход по списку. Если текущий элемент списка – это подстрока, не содержащая помеченных элементов, то она помещается в выходной набор *Tiles*, а все элементы этой подстроки помечаются в *P* и *T*. Если длины подстрок в списке (*MaxMatch*) больше значения *MinimumMatchLength*, то происходит возврат к первому этапу.

К достоинствам алгоритма можно отнести игнорирование им строк более коротких, чем *MinimumMatchLength*, что позволяет пропускать небольшие совпавшие части и сокращать время поисков. Но при неверно заданном пороговом значении это может стать недостатком, так как могут быть пропущены значимые фрагменты. Сложность алгоритма может достигать порядка $O(n)$.

Метод идентификационных меток. При поиске конфиденциальной информации речь может идти о работе с текстами большего объема. В этом случае непосредственное сравнение исходного файла с базой шаблонов может быть не эффективно.

Данная техника позволяет перевести файл (документ) в более краткое представление: документу сопоставляется набор идентификационных меток (fingerprints), так чтобы для близких документов эти наборы пересекались.

Исходный текст разбивается на *k*-граммы – любые *k* символов стоящих подряд.

Количество k-граммов, которые можно построить для текста длины *l* обозначается π, $\pi=(1-(k-1))$. Далее все k-граммы хешируются. Получившийся набор хеш-значений (h_1, h_n)характеризует исходный документ.

На практике, использовать все значения не целесообразно, поэтому выбирают небольшое их подмножество. Выбранные хеш-значения становятся метками (*fingerprints*) документа. Вместе с самой меткой хранится информация о том, какому файлу она принадлежит, и в каком месте этого файла встречается. Если хеш-функция гарантирует малую вероятность коллизий, то одинаковая метка в наборах двух файлов свидетельствует о том, что у них есть общая подстрока. По количеству общих меток можно судить о близости файлов. Чтобы выбрать те хеш-значения, которые будут представлять документ, используют следующие подходы:

1. Наивный подход – выбирается каждое *i*-тое из π значений.
2. По Хайнце следует назначать метками минимальные хеш-значения, их количество для всех документов будет постоянно.
3. Манбер предложил выбирать в качестве меток только те хеш-значения, для которых $h = 0 \bmod p$, так останется только π/p меток.
4. Метод просеивания (*winnowing*) гарантирует, что если в двух файлах есть хотя бы одна достаточно длинная общая подстрока, то как минимум одна метка в их наборах совпадет.

К достоинствам алгоритма можно отнести игнорирование случайно совпавших подстрок и линейной трудоемкости. В то же время, неверно заданные параметры могут серьезно повлиять на результаты в негативном ключе.

LCS алгоритм. Рассмотрим оригинальный LCS алгоритм, используемый для сравнения двух строк [2].

Пусть даны строки $X=x_1x_2\ldots x_n$, $Y=y_1y_2\ldots y_m$, $Z=z_1z_2\ldots z_p$. Строка Z является общей подпоследовательностью, если для ее представления из строки X могут быть удалены n-p символов, а из строки Y – m-p символов. Таким образом, задача LCS алгоритма – поиск наибольшей строки Y, являющейся общей подстрокой для X и Y.

Пусть имеется массив префиксов, заканчивающихся символами с номерами i и j, последовательностей X и $Ypref[i][j]$. Получается следующее рекуррентное соотношение:

$$pref = \begin{cases} 0, i = 0 \text{ или } j = 0 \\ pref[i-1][j-1]+1, x[i] = y[j] \\ \max\left(pref[i][j-1], pref[i-1][j], x[i] \neq y[j]\right) \end{cases}$$

Недостатком данного алгоритма является его возможная неточность при сравнении двух строк, вероятность пропусков символов или же, наоборот, нечувствительность к их удалению. Кроме того, данный алгоритм имеет сложность $O(n*m)$.

2. Анализ алгоритма Хескела

Среди рассмотренных алгоритмов наибольший интерес вызывают LCS алгоритмы в силу возможности их модификации. Один из вариантов был предложен Паулом Хескелем для сравнения двух файлов [3]. В качестве условной единицы будет использоваться строка файла.

Пусть есть два файла *O* и *N*, будем сравнивать файл *N*с файлом O*,* однако алгоритм симметричный и может работать и наоборот. Текст строки файла является ключом для подсчета количества вхождений. Для этого будут использоваться два счетчика: *OC* и *NC* для хранения количества вхождений строки в файлы *O* и *N* соответственно. Также, пусть есть таблица *T* и два массива *OA* и *NA*. Таблица *T* имеет поле *OLNO*, содержащее номер строки в файле *O*. Массивы *OA*имеет запись для каждой строки файла *O* и содержит или указатель на строку в таблице *T,* или номер строки в файле *N*.

Алгоритм работает за 6 проходов.

1. Обработка файла *N*:

– Каждая *i-ая* строка файла *N* считывается как последовательность;

– Строка добавляется в таблицу *T*, если она еще в ней не содержится;

– Увеличивается счетчик *NC* для данной строки;

– В *NA*[*i*] устанавливается указатель на строки в таблице символов.

2. Обработка файла *O*:

– Каждая *i-ая* строка файла *N* считывается как последовательность;

– Строка добавляется в таблицу *T*, если она еще в ней не содержится;

– Увеличивается счетчик *OC* для данной строки;

– В *OA*[*i*] устанавливается указатель на строки в таблице *T;*

– Поле *OLNO* заполняется номер строки.

3. Действия выполняются только для строк, для которых выполняется равенство *NC=OC=1*, то есть для уникальных. Для каждой такой строки мы заменяем указатели *NA* и *OA* на номер строки в другом файле. Например, пусть есть строка *NA*[*i*], тогда найдем строку в таблице *T* и установим *NA*[*i*] и *OA*[*OLNO*] равным *i*.

4. Происходит обработка каждой строки в *NA*следующим образом: если *NA*[*i*] указывает на *OA*[*j*] и *NA*[*i*+1] и *OA*[*j*+1] содержат идентичные указатели, тогда *NA*[*i*+1] задается *j*+1, а *OA*[*j*+1] – *i*+1.

5. Действия производятся «в другую сторону»: если $NA[i]$ указывает на $OA[j]$ и $NA[i\text{-}1]$ и $OA[j\text{-}1]$ содержат идентичные указатели, тогда $NA[i\text{-}1]$ задается j-1, а $OA[j\text{-}1] – i$-1.

6. В последнем проходе происходит описание всех обработанных строк. Теперь NA содержит следующую информацию: если $NA[i]$ указывает на вхождение строки в таблицу символов, значит, строка i не содержится в файле O. Если $NA[i]$ указывает на $OA[j]$, но $NA[i+1]$ не указывает на $OA[j+1]$, значит, строка i является границей, строки после которой не содержатся в N.

Несмотря на необходимость 6 проходов, алгоритм имеет линейную сложность.

3. Модификация алгоритма Хескела

Рассмотрим алгоритм Хескела в контексте решаемой задачи поиска конфиденциальной информации по базе шаблонов.

В первую очередь, будет происходить обработка лишь одного рассматриваемого файла. Кроме того, используемая при работе с оригинальным алгоритмом структурная единица в виде строки является неподходящей, так как является слишком большим элементом. Вместо этого будут использоваться k-граммы.

Таким образом, мы имеем текст в виде строки S, а также базу данных шаблонов BD. Пусть имеется таблица для хранения k-граммов и счетчиков количества их появления.

Рассмотрим проходы, которые будут использоваться:

1. Работа с исходным текстом:
   – Разбиение текста на k-граммы фиксированной длины;
   – Добавление каждого k-грамма в таблицу, если он в ней еще не содержится;
   – Увеличение счетчика для каждого k-грамма.
2. Рассматриваются только уникальные k-граммы, для которых счетчик равен 1. Для каждого из них происходит поиск по базе шаблонов.
3. Если найдены совпадения, осуществляется проверка предшествующих и последующих k-граммов.

Таким образом, оригинальный алгоритм Хескела может быть сокращен всего до трех проходов.

Кроме того, структура алгоритма позволяет использовать методы хеширования для обеспечения дополнительной защиты информации. Операция хеширования применяется к исходному тексту после его разбиения на k-граммы. В случае же базы шаблонов, возможно как и хранение в хешированном виде, так и обработка непосредственно перед операцией сравнения.

Заключение

В данной статье рассмотрены основные алгоритмы, которые могут быть использованы для обнаружения текстуально совпадающей информации. В качестве основы для работы была выбрана группа LCS алгоритмов.

После изучения различных вариантов был выбран и рассмотрен алгоритм Паула Хескела. На его основе был разработан модифицированный алгоритм.

Полученный алгоритм будет использован при разработке комплекса программных средств, осуществляющих поиск конфиденциальной информации в текстовых документах.

Список литературы

1. Regnier, M. Knuth-Morris-Pratt Algorithm: An Analysis / M. Regnier // MFCS J89. Mathematical foundations of computer science. – Poland, 1989. – P. 431-444.

2. Bergroth, L. A survey of longest common subsequence algorithms / L. Bergroth, H. Hakonen, T. Raita // Proceedings Seventh International Symposium on String Processing and Information Retrieval. – Spain, 2000. – P. 39-48.

3. Heckel, P. A Technique for Isolating Differences Between Files / P. Heckel // Communications of the ACM. – 1978. – Vol. 21 (4). – P. 264-268.

Защита данных в базах данных от несанкционированного доступа средствами СУБД

М. А. Грибкова

Студент магистрант

В. А. Степанцов

Доцент

Введение

В настоящее время практически все сферы жизни человека связаны с доступом и использованием электронных данных. Одновременно с увеличением информации в цифровом формате все более актуальным становился вопрос возможности ее хранения. И основным решением этой проблемы стало использование баз данных, которые, кроме

непосредственно хранения информации, предоставляют инструменты для ее обработки.

Однако, такие хранилища данных, собранных в одном месте, которые, зачастую, могут нести какую-либо ценность, привлекают не только пользователей, но и злоумышленников. Современные базы данных, несмотря на постоянные обновления и попытки сделать их более совершенными, удобными и защищенными, содержат множество уязвимостей, которые могут быть использованы для получения несанкционированного доступа.

Таким образом, проблема обеспечения безопасности информации в базах данных является достаточно актуальной. В рамках данной работы будут рассмотрены методы защиты информации в СУБД с помощью встроенных средств, а именно, с помощью использования функции журналирования.

1. Журналирование

Журналирование – встроенная функция СУБД, которая позволяет сохранять информацию, необходимую для восстановления базы данных в предыдущее состояние в случае каких-либо сбоев, путем отслеживания различных событий, происходящих в ней.

Журналирование помогает решать ряд задач:

1. Отслеживание кто, когда, с какой машины и с помощью какого приложения имел доступ к базе или производил какие-либо изменения. В случае обнаружения нарушения целостности или конфиденциальности хранимой информации журнал может быть использован для поиска злоумышленника.

2. Ведение истории изменений данных, то есть, своего рода поддержка контроля версий. Может использоваться в случае необходимости хранения всех изменений, производимых с информацией, старых и новых значений. Срок хранения может варьироваться как от нескольких дней, так до всего срока жизни СУБД.

Кроме этих двух задач, естественно, существует еще огромное количество вопросов, которые могут быть решены с применением функции журналирования, например, поддержка уведомлений об изменениях данных или синхронизация обновления данных на разных серверах. Однако стоит сосредоточиться именно на двух вышеперечисленных, так как они относятся непосредственно к интересующей проблеме обеспечения безопасности информации.

В данной работе будет рассматриваться использование функции журналирования на примере СУБД Microsoft SQL Server.

2. Средства журналирования Microsoft SQL Server

Microsoft SQL Server имеет стандартное встроенное средство для мониторинга событий – SQL Server Profiler [1].

SQL Server Profiler – графический пользовательский интерфейс для создания трассировок, управления ими, а также анализа и воспроизведения их результатов. Полученные результаты могут сохраняться как в сторонний файл, так и непосредственно в таблицу базы данных. Данное приложение может быть использовано для ряда следующих задач:

– Отслеживание всех обрабатываемых sql-запросов;

– Пошаговое выполнение sql-запросов для поиска источников проблемы в случае их некорректной работы;

– Выявление и диагностика медленно работающих запросов, влекущих за собой снижение производительности всей системы;

– Анализ общей производительности системы;

– Отслеживание происходящих в системе событий;

– Моделирование и диагностика проблем сервера;

– Аудит действий, связанных с безопасностью, для дальнейшего контроля администратором безопасности.

Рассмотрим основные понятия.

Трассировка – деятельность системы, происходящая в фоновом режиме на машине Microsoft SQL Server, которая перехватывает все или определенные события и данные, передаваемые или обрабатываемые этими событиями. Файлы трассировок создаются и поддерживаются посредством языка Transact-SQL (T-SQL). Однако так же может быть использован и графический интерфейс SQL Server Profiler, что позволяет существенно упростить процесс работы.

Событие – любое действие в рамках СУБД, например, наиболее часто встречающиеся это:

– Подключение пользователей, сбои, отключения;

– Проверка прав доступа;

– Операции SELECT, UPDATE, INSERT, DELETE;

– Запись ошибок и другие.

Источником события может быть любой источник, вызывающий событие трассировки. Все данные, создаваемые событием, отображаются в трассировке одной строкой.

Класс события (Event Class) – тип события трассировки, который содержит все данные, которые может сообщить это событие, например, Object:Created – событие создания объекта.

Кроме того, при создании трассировки могут быть настроены фильтры, которые позволяют определять критерии для фильтрации

данных, собираемых событием. Например, можно ограничить пользователей, информация о действиях которых должна присутствовать в данной трассировке.

3. Формулировка задачи

Сформулируем задачу.

Пусть необходимо создать базу данных, соответствующую следующим требованиям (рис. 1):

1. База данных содержит набор шаблонов для использования сторонним приложением.

2. Существует пользователь «Администратор», который может добавлять пользователей «Экспертов», назначать уровень доступа существующим и удалять их.

3. Пользователи «Эксперты» могут добавлять шаблоны и редактировать или удалять существующие шаблоны.

4. Для работы пользователям «Экспертам» предоставляется таблица-копия основной таблицы, содержащей шаблоны, то есть все действия совершаются с копией, а не оригиналом.

5. Пользователь «Администратор», может подтверждать или отклонять действия пользователей, совершенных в рамках 2 пункта.

6. В соответствии с разрешением или запретом пользователя «Администратора» изменения, произведенные в таблице-копии должны применяться или не применяться к таблице-оригиналу.

Для поддержки 5 требования и будет использоваться функция журналирования. Для этого необходимо отслеживать, какие данные и каким пользователем «Экспертом» были изменены, добавлены или удалены.

4. Настройка трассировок SQL Server Profiler

Одним из наиболее важных моментов работы с трассировками является их настройка. Microsoft SQL Server поддерживает сотни различных событий, однако использование всех их может привести к снижению скорости работы СУБД, а также в разы усложнить поиск необходимой информации.

В рамках сформулированной задачи необходимо получать следующую информацию:

- Имя пользователя;
- Имя рабочей станции;
- Идентификатор измененного объекта;
- Произведенные изменения;
- Время и дата совершения изменений.

Рис. 1. Диаграмма вариантов использования

С учетом уровня доступа, который будут иметь пользователи «Эксперты», им будет доступно совершение операций INSERT, UPDATE, DELETE. Значит, необходимо настроить трассировку, которая будет отслеживать именно эти действия.

Для этого может использоваться категория событий T-SQL, а именно класс событий SQL:StmtCompleted, который указывает, что завершено выполнение инструкции языка T-SQL.

Сначала создается трассировка, в рамках данной задачи все данные, полученные при ее выполнении, будут сохраняться в таблицу TrackChanges самой базы данных (рис. 2).

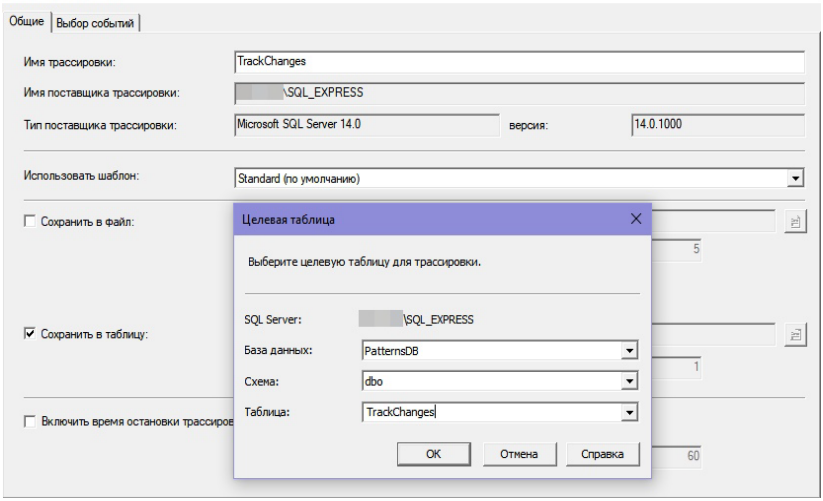


Рис. 2. Создание трассировки

Далее необходимо выбрать класс событий и параметры, которые необходимо сохранять (рис. 3).

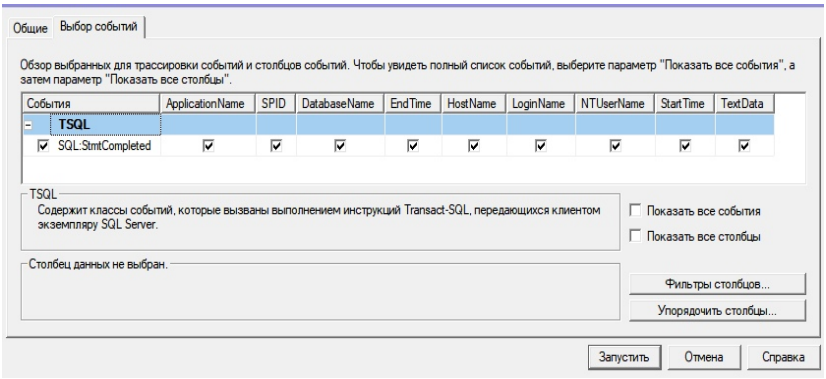


Рис. 3. Выбор класса событий

При запуске трассировки и выполнении простого запроса INSERT, в журнале появляется множество записей (рис. 4).

DatabaseName	LoginName	NTUserName	TextData	StartTime	EndTime
PatternsDB	MicrosoftAccount\mar...	marusya...	exec master.dbo.xp_instance_regread ...	2020-05-05 15:25:37...	2020-05-05 15:25:37...
PatternsDB	MicrosoftAccount\mar...	marusya...	BEGIN TRY	2020-05-05 15:25:37...	2020-05-05 15:25:37...
PatternsDB	MicrosoftAccount\mar...	marusya...	SELECT @cluster_name = cluster_name,...	2020-05-05 15:25:37...	2020-05-05 15:25:37...
PatternsDB	MicrosoftAccount\mar...	marusya...	END TRY	2020-05-05 15:25:37...	2020-05-05 15:25:37...
PatternsDB	MicrosoftAccount\mar...	marusya...	SELECT @SmoAuditLevel AS [AuditLevel...	2020-05-05 15:25:37...	2020-05-05 15:25:37...
PatternsDB	MicrosoftAccount\mar...	marusya...	SELECT @@SPID	2020-05-05 15:25:46...	2020-05-05 15:25:46...
PatternsDB	MicrosoftAccount\mar...	marusya...	USE [PatternsDB]	2020-05-05 15:25:46...	2020-05-05 15:25:46...
PatternsDB	MicrosoftAccount\mar...	marusya...	INSERT INTO [dbo].[WorkCopy] ...	2020-05-05 15:25:46...	2020-05-05 15:25:46...
PatternsDB	MicrosoftAccount\mar...	marusya...	use [PatternsDB]	2020-05-05 15:25:58...	2020-05-05 15:25:58...
PatternsDB	MicrosoftAccount\mar...	marusya...	select @HkeyLocal=N'HKEY_LOCAL_MACHINE'	2020-05-05 15:26:05...	2020-05-05 15:26:05...
PatternsDB	MicrosoftAccount\mar...	marusya...	select @MSSqlServerRegPath=N'SOFTWAR...	2020-05-05 15:26:05...	2020-05-05 15:26:05...
PatternsDB	MicrosoftAccount\mar...	marusya...	select @InstanceRegPath=@MSSqlServer...	2020-05-05 15:26:05...	2020-05-05 15:26:05...
PatternsDB	MicrosoftAccount\mar...	marusya...	select @FilestreamRegPath=@InstanceR...	2020-05-05 15:26:05...	2020-05-05 15:26:05...

Рис. 4. Журнал трассировки

Среди данных записей можно увидеть строку выполнения запроса INSERT, о котором можно просмотреть подробную информацию: данные, которые были вставлены, кем и во сколько (рис.5).

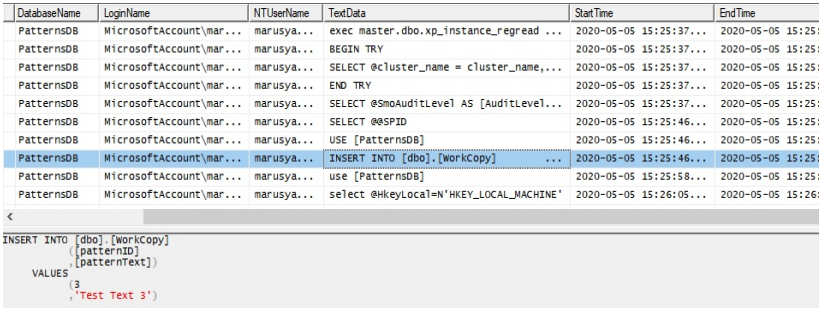


Рис. 5. Подробная информация о выполнени запрос INSERT

Аналогичным образом выглядят и записи о выполнении запросов UPDATE и DELETE.

5. Построение базы данных

Таким образом, для покрытия требований можно построить базу данных, состоящую из следующих таблиц (рис. 6):

– Users – таблица, содержащая логин, пароль и уровень доступа пользователя;

– Patterns – таблица для хранения шаблонов, информации о том кем и когда они были добавлены и/или изменены последний раз;

– WorkCopy – таблица, с которой работают пользователи «Эксперты». В момент открытия содержит последние актуальные значения таблицы Patterns;

– TrackChanges – системная таблица, генерируемая и обновляемая SQL Server Profiler;

– Updates – таблица, заполняемая сторонним приложением на основании таблицы TrackChanges. На ее основании пользователем «Администратором» производится проверка обновлений, внесенных пользователями «Экспертами». Содержит идентификатор шаблона, действие, которое было с ним произведено (UPDATE, INSERT, DELETE), старый и новый текст шаблона, а также кем и когда были произведены изменения.

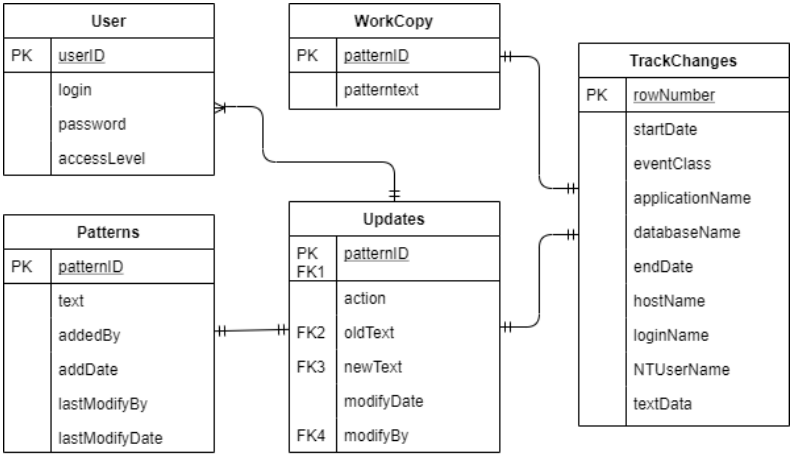


Рис. 6. Схема базы данных

Таким образом, у пользователей «Экспертов» нет доступа непосредственно к базе шаблонов, все их действия отслеживаются и регистрируются в системе. Даже при возникновении факта несанкционированного доступа, всегда будет иметься информации о том, с какой машины и какой пользователь совершил злоумышленные действия.

Заключению

В рамках данной работы был рассмотрен способ защиты информации в СУБД Microsoft SQL Server с помощью стандартного встроенного средства для журналирования – SQL Server Profiler.

Была рассмотрена функция трассировки, доступная в данном программном продукте. И на примере поставленной задачи был показан пример настройки трассировок для отслеживания внесения изменений в информацию, хранящуюся в базе данных. Кроме того, была построена схема базы данных, позволяющая разграничить доступ различных

категорий пользователей к данным с целью обеспечения дополнительной безопасности.

Полученные результаты будут использованы для разработки комплекса программных средств, осуществляющих поиск конфиденциальной информации в текстовых документах.

Список литературы

1. Microsoft SQL Documentation [Электронный ресурс] : техническая документация. – Режим доступа : https://docs.microsoft.com/en-us/sql

Методы принятия решений при планировании технологических карт в растениеводстве

Е. В. Григорова

Студент магистрант

И. В. Илларионов

Доцент

Введение

В современном мире сельское хозяйство является одним из основных направлений деятельности человека. Одной из главных отраслей сельского хозяйства является растениеводство. Информация о технологическом процессе выращивания сельскохозяйственных культур фиксируется в технологических картах. При их составлении невозможно учесть все негативные факторы, влияющие на качество урожая. Современные математические методы принятия решений могут упростить соблюдение плана выполнения работ по выращиванию культур в соответствии с технологическими картами и внешними факторами.

Целью данной работы является анализ технологических карт и возможность применения методов принятия решений при планировании и дальнейшем использовании технологических карт в сельском хозяйстве.

1. Анализ предметной области

Выращивание сельскохозяйственных культур представляет собой последовательность выполняемых операций. При этом, каждая

последующая операция зависит от результата выполнения предыдущей. Совокупность выполняемых операций, а также ресурсы являются технологией производственного процесса, которую, в свою очередь, можно представить в виде технологической карты.

Технологическая карта содержит в себе перечень агротехнических операций, способствующих увеличению количества и качества урожая. Каждую операцию выделяют в отдельный элемент технологической карты.

Технологическая карта разрабатывается для каждой конкретной культуры на один год. Для ее составления необходимо определить:

1. Структуру посевных площадей, а также площадь обрабатываемых полей;
2. Парк агрегатов, пригодных для возделывания сельскохозяйственной культуры;
3. Остатки материалов на складах, для планирования затрат;
4. Фонд работников организации, проанализировать количество трудоспособных сотрудников;
5. Нормы выработки для механизированной и ручной работы, а также определить расценки выполнения каждой единицы выполненной работы;
6. Перечень необходимых для возделывания культуры материалов;
7. Потребность использования услуг сторонних организаций, а также их стоимость и доступность;
8. Проанализировав объемы урожайности предыдущих периодов (если такие данные присутствуют), определить объем урожайности будущего периода с учетом корректировки структуры посевных площадей [1].

Наглядное отображение структуры технологической карты в области растениеводства, а также взаимосвязь составляющих технологической карты и последовательность выполнения операций представлена на рис. 1.

2. Выявление показателя урожайности с помощью Байесовских сетей

В качестве метода принятия решений воспользуемся байесовскими сетями для представления взаимосвязей между объясняющими переменными и урожайностью сельскохозяйственной продукции и изучение ожидаемой урожайности этой продукции при разнообразных условиях.

Байесовская сеть, или байесовская сеть доверия (БС), представляет собой совместное распределение вероятностей множества случайных величин, причинные связи между которыми представлены в виде

направленного ациклического графа. Каждая случайная переменная имеет, как правило, дискретный набор состояний, образующих множество случайных событий.

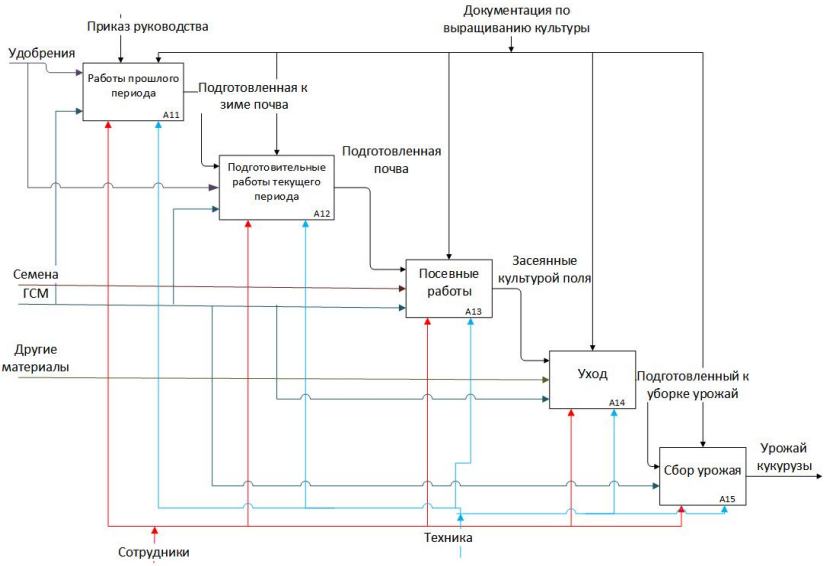


Рис. 1. Диаграмма системы «Выращивание кукурузы на зерно»

Таким образом, БС состоит из качественной части направленного графа, представленной в графической форме, и количественной части (таблицы условных вероятностей), отражающей реакции в дочерних узлах на изменение в причинных узлах сети. Математически БС описывает зависимости между условными вероятностями множества случайных переменных [2-3].

В связи с этим БС представляются наиболее подходящий технологией анализа детерминант урожайности сельскохозяйственной продукции

3. Выбор переменных и их дискретизация

В качестве целевого узла выступила урожайность кукурузы (ц/га) в 2019 г. Переменные, которые вошли в итоговую модель, включают:

– параметры агроэкологических условий производства (сумма осадков, степень засушливости климата);

– структурные характеристики предприятий (специализация производства, посевная площадь, доля кукурузы в структуре посевных площадей);

– элементы агротехнологии (внесение удобрений, применение средств защиты растений);

– технические возможности предприятия (наличие новых комбайнов, тракторов и посевных агрегатов).

Одним из аспектов разработки БС является возможность дискретизации переменных, т. е. преобразования переменных, измеренных по метрической шкале, в категориальные, в свою очередь, измеренных по порядковой шкале. Дискретизация предполагает выбор количества категорий и их пороговых (граничных) значений. Понимание свойств переменной должно давать возможность качественной экспертной дискретизации, особенно выбора пороговых значений.

4. Результат применения Байесовской сети

Конечная сеть содержит 9 переменных и целевую переменную «Урожайность кукурузы (2019 г.)», которая представлена на рис. 2.

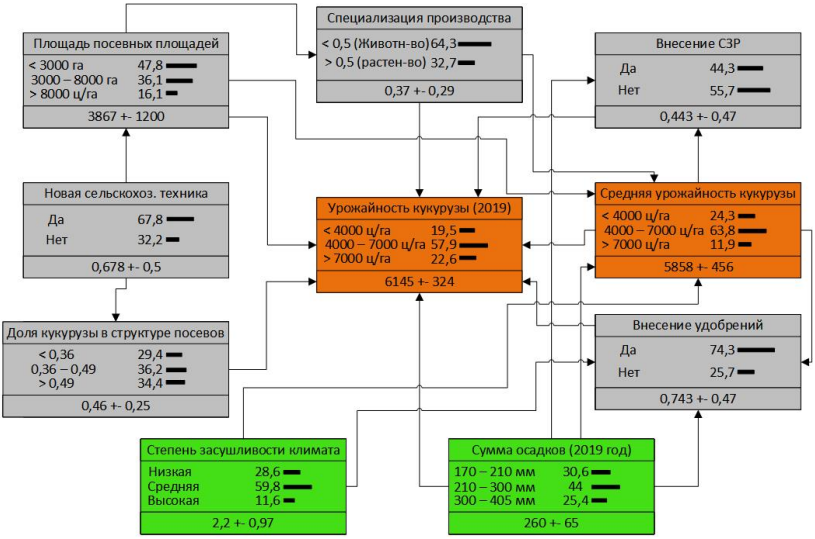


Рис. 2. Байесовская сеть детерминант урожайности кукурузы

Анализ связей показал, что урожайность кукурузы в 2019 г. напрямую зависит от средней урожайности культуры в

предшествующий период. Множество переменных, имеющих прямую связь с целевым узлом, включает:

– Сумма осадков (2019), мм;
– Площадь посевных площадей, га;
– Доля кукурузы в структуре посевов;
– Специализация производства (растениеводство или другое);
– Внесение удобрений (да или нет);
– Внесение средств защиты растений (да или нет).
– Средняя урожайность кукурузы зависит в БС от:
– Специализации производства;
– Площади посевных площадей;
– Степени засушливости климата;
– Суммы осадков.

При этом средняя урожайность явилась определяющим фактором внесения СЗР и удобрений.

На рис. 2 показана БС с начальными, априорными вероятностями случайных событий отобранных переменных, основные из них:

– Только 22,6% сельскохозяйственных предприятий имели урожайность более 7000 ц/га в 2019 году, в свою очередь, 19,5% предприятий получили урожайность менее 4000 ц/га;

– Более 71% организаций ведут свою деятельность в условиях средней и высокой засушливости;

– Чуть меньше 48% организаций являются малыми предприятиями с малой площадью посевных площадей;

– Большая часть предприятий имеет в своем распоряжение современную сельскохозяйственную технику;

– Около 56% предприятий использовали средства защиты растений;

– Практически 75% сельскохозяйственных организаций вносили удобрения в 2019 году;

– В основном организации имеют прибыль от реализации продукции животноводства, а не от реализации продукции растениеводства.

Заключение

Данная статья ориентирована на изучение детерминант урожайности. Данное исследование представляет собой анализ детерминант урожайности кукурузы среди сельскохозяйственных предприятий. Для этого была разработана байесовская сеть, которая оказалась гибким инструментом, что позволило объединить качественную информацию с количественными данными. Комбинация графического интерфейса с эмпирическим анализом позволила

включить качественные экспертные знания заинтересованных сторон с количественными данными. Результаты подчеркивают необходимость совершенствования управленческих решений в области ведения сельского хозяйства, что должно способствовать выработке политики, направленной на повышение урожайности сельскохозяйственных культур. Такие практики могли быть направлены на строгое соблюдение существующих рекомендаций по технологии выращивания сельскохозяйственных культур, а также в стимулирование развития экологически чистого сельского хозяйства.

Список литературы

1. Яковлев, Б. И. Организация производства и предпринимательства в АПК. Учебник / Б. И. Яковлев, В. Б. Яковлев. – СПб. : Квадро, 2016. – 480 с.

2. Ларичев, О. И. Теория и методы принятия решений / О. И. Ларичев. – М. : Логос, 2000. – 296 с.

3. Тулупьев, А. Л. Циклы в байесовских сетях : вероятностная семантика и отношения с соседними узлами / А. Л. Тулупьев, С. И. Николенко, А. В. Сироткин // Тр. СПИИРАН. – 2006. – Вып. 3. – № 1. – С. 240-263.

Разработка биометрической системы для аутентификации пользователя по клавиатурному почерку

Е. С. Данковцева

Студент магистрант

Е. Ю. Митрофанова

Доцент

Введение

Авторизация и аутентификация являются неотъемлемой частью нашей жизни. Существует несколько традиционных способов аутентификации личности: по знаниям, по собственности, по биометрическим параметрам. Но использование только какой-либо собственности или секретной информации не является абсолютной гарантией того, что посторонний не сможет получить конфиденциальную информацию или чужую собственность.

После входа в систему пользователь может отлучиться по каким-то делам от рабочего места на некоторое время. И как часто бывает, он может не заблокировать свой компьютер. В этот момент другой человек может воспользоваться ПК. Например, он может изменить какой-то важный документ, просмотреть файлы или почту, скопировать или удалить что-то.

В соответствии с этим было принято решение о разработке системы, которая могла бы в скрытом режиме идентифицировать постороннего пользователя по клавиатурному почерку. Предполагается, что в случае обнаружения постороннего, компьютер будет заблокирован, чтобы злоумышленник не мог иметь доступ к устройству. Использование подобных продуктов должно повысить надежность систем, но при этом не возникает необходимости покупки какого-либо дополнительного оборудования [1, 2].

1. Клавиатурный почерк

Если рассматривать скрытую верификацию пользователя по свободному тексту, то основной сложностью является ввод пользователем любой последовательности символов, к тому же пользователь может на время прерываться, задумываться, делать опечатки. Не представляется возможным анализировать текст по словам или фразам, так как нужно будет хранить много данных, а постоянная обработка введенного теста в режиме реального времени будет давать больную вычислительную нагрузку. Поэтому при скрытой верификации происходит анализ введенных диграфов.

Диграф – это сочетание двух последовательно идущих в тесте символов.

В труде Гейнса [3] описано тестирование системы по распознаванию почерка, в котором участвовало шесть профессиональных секретарей. Тестирование проводилось дважды с периодом в четыре месяца. В данной работе анализируется характеристики клавиатурного почерка участниц с помощью t-критерия Стьюдента, а также используются коэффициент асимметричности и коэффициент эксцесса. В итоге авторы пришли к выводу, что характеристики клавиатурного почерка распределены нормально. Также отмечается, что характеристики, полученные через четыре месяца, мало отличались от тех, что получены в первый раз. Это указывает на то, что клавиатурный почерк является довольно стабильным биометрическим параметром, по которому возможно верифицировать пользователя.

В исследовании Савинова [4] говорится о том, что характеристики клавиатурного почерка на самом деле можно характеризовать как бимодальное распределение. Дело в том, что есть два разных варианта

одного и того же диграфа: с перекрытием и без него. Анализируя данный факт, автор пришел к выводу, что диграфы без перекрытий распределены по нормальному закону, а также и диграфы с перекрытиями распределяются по нормальному закону. В соответствии с этим по интервалу между символами можно выделить два случая диграфа и рассматривать каждый по отдельности.

2. Аутентификация пользователя

При регистрации в системе происходит фиксирование каждого события нажатия и отпускания клавиши клавиатуры. После получения каждого события клавиатуры, происходит вычисление времени удержания клавиши и временной промежуток между соседними клавишами.

Для каждого такого временного отрезка может быть рассчитан интервал изменения параметра $\left[\min(v_i), \max(v_i)\right]$, который будет в дальнейшем использован при верификации пользователя.

Максимальное и минимальное значение для параметра v_i вычисляется, как:

$$\min(v_i) = m(v_i) - C[L,(1-P_i)] \times \sigma(v_i), \quad (1)$$

$$\max(v_i) = m(v_i) + C[L,(1-P_i)] \times \sigma(v_i). \quad (2)$$

где L – число использованных примеров при обучении, $C[L,(1-P_i)]$ – коэффициенты Стьюдента, P_i – заданное значение вероятности ошибок первого рода (ошибок ложного отказа).

У каждого диграфа есть четыре параметра, для каждого из которых рассчитывается в дальнейшем интервал изменения параметра:

- время удержания первого символа t_1;
- время удержания второго символа t_2;
- время между первым и вторым символом t_3;
- длительность всего диграфа t_4.

В соответствии с этим, для каждого такого параметра определены ошибки E_i как общее количество непопаданий t_i в соответствующий интервал. Для каждого экземпляра диграфа d_i происходит проверка всех параметров t_i и вычисляются ошибки E_i. Далее анализируется отношение ошибки к общему количеству n всех представленных экземпляров диграфов d_i.

$$\frac{E_i}{n} < k \,. \tag{3}$$

Значение данного коэффициента k было определено экспериментально.

Если хотя бы одно неравенство из четырех не выполняется, то считается, что данный диграф вводился неправильно или был введен чужим пользователем. Такой диграф помечается в системе как «неправильный».

Обозначим общее количество «неправильных» диграфов N_w а количество всех представленных системе диграфов N.

Тогда можно определить решающее правило следующего вида:

$$\frac{N_w}{N} < K \,. \tag{4}$$

Значение K определяется для каждого пользователя во время регистрации на основе общего количества «неправильных» диграфов, введенных самим пользователем.

Если в ходе верификации данное неравенство не выполняется, то принимается решение, что системой в данный момент пользуется злоумышленник.

3. Реализация

В качестве средств реализации использовались:

– Microsoft Visual Studio Enterprise 2017;
– Visio Professional 2016.

В системе будут регистрироваться пользователи с разными логинами, поэтому используемая система является верификационной.

Когда приложение запущено, пользователь может выбрать два основных режима работы: регистрация и верификация (рис. 1). Режим регистрации включает в себя возможность дообучения системы для пользователя, который уже существует в базе данных.

Когда пользователь нажал на кнопку «Start», приложение начинает работать в фоновом режиме и его можно свернуть. Далее в любом месте пользователь должен набирать текст так, как он это делает обычно.

Как только пользователь наберет определенное количество символов, регистрация будет закончена, и вся информация о клавиатурном почерке будет сохранена в базу данных.

Далее можно проверить работу приложения по верификации по биометрическому параметру. Для этого нужно ввести имя пользователя, за которого он хочет себя выдать, и выбрать соответствующий Radiobutton на экране.

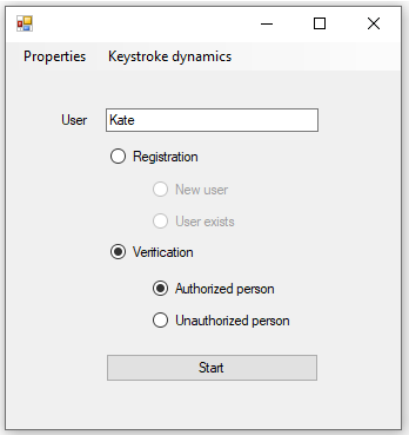


Рис. 1. Главное окно приложения

После прохождения верификации для пользователя доступна функция анализа клавиатурного почерка в верхнем меню. При нажатии на кнопку «Keystroke dynamics» появляется новое окно, где можно выбрать параметр, который будет показан пользователю. При этом можно открыть характеристики по всем параметрам одновременно (рис. 2).

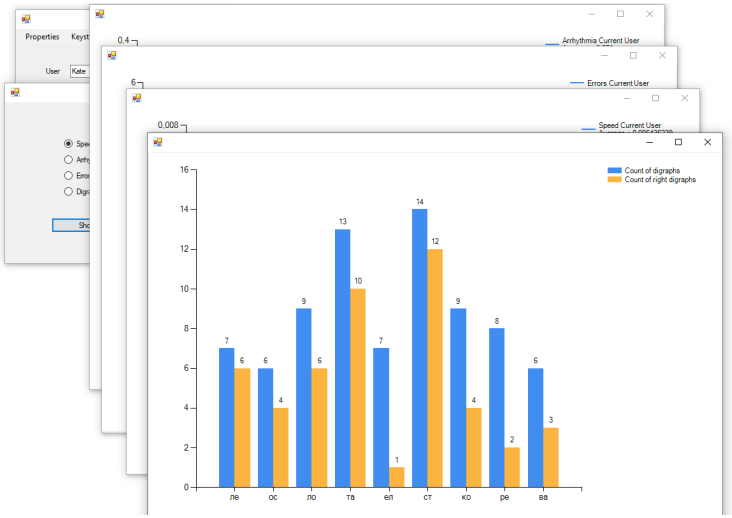


Рис. 2. Окна для анализа клавиатурного почерка

Заключение

Было разработано приложение, которое в скрытном режиме может верифицировать пользователя по клавиатурному почерку. Метод аутентификации был разработан самостоятельно на основе метода измерения близости образа к биометрическому эталону мерой Хэмминга и на основе анализа данных о клавиатурном почерке, которые были получены экспериментально.

Также доступен анализ клавиатурного почерка пользователя после верификации в системе.

Список литературы

1. Лакин, Г. Ф. Биометрия : Учебное пособие для биол. спец. вузов. – 4-е изд., перераб. и доп. – М. : Высшая школа, 1990. –352 с.

2. Руководство по биометрии / Р. М. Болл [и др.]; пер. с англ. Н. Е. Агаповой. – М. : Техносфера, 2007. –368 с.

3. Authentication By keystroke timing: some preliminary results / R. Stockton Gaines, W. Lisowski, S.J. Press, N. Shapiro. – Santa Monica : Rand, 1980.

4. Говорящие клавиши. Характер человека определят по клавиатурному почерку [Электронный ресурс] : Электрон. Журн. – Режим доступа : http://smartnews.ru/regions/yoshkarola/17278.html

Имитационная модель структуры полимеров

С. А. Зайцев

Студент магистрант

М. Е. Семенов

Профессор

Введение

Развитие современной промышленности характеризуется быстрым ростом объема производства и сбыта пластмасс и каучуков. Таким образом, учитывая общемировую тенденцию на увеличение потребления полимеров, в частности синтетических каучуков, одним из приоритетов как для общемирового, так и для отечественного производителя становится интенсификация производства, путем разработки и внедрения инновационных технологий.

Создаются новые способы получения пластических масс с необходимым комплексом эксплуатационных характеристик, например, применяется ионизирующее излучение. Данная методика также позволяет решить проблему переработки и утилизации вышедших из эксплуатации пластмассовых изделий. В частности, бутилкаучук, подвергнутый радиационной регенерации, представляют собой ценное органическое сырье.

Однако, в связи с высокой технологичностью, данная методика не является массовой. Для внедрения ее в производство необходимо разработать математико-программные средства моделирования, позволяющие строить пространственные модели внутренних структур различных полимерных материалов и достоверно моделировать процессы, происходящие в них при радиолизе.

1. Строение полимеров

Макромолекула полимера в большинстве случаев состоит из многочисленных повторяющихся элементарных звеньев, связанных жесткой связью, т. е. мономеров. Конформация и конфигурация мономеров во многом определяют длину цепи макромолекулы, способ её построения и взаимное расположение атомов и групп атомов в цепи [1].

Характеристиками, описывающими внутреннюю пространственную структуру полимеров, как в расплаве и растворе, так и в кристаллическом состоянии, являются угол внутреннего вращения – угол поворота одних групп атомов молекулы относительно других – и валентный угол – угол, образованный направлениями химических связей, исходящими из одного атома. Случай карбоцепного полимера полиэтилена представлен схематически на рис. 1.

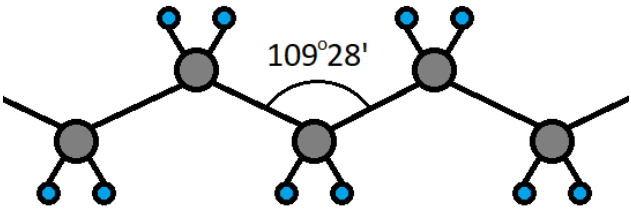


Рис. 1. Фрагмент молекулы полиэтилена с валентным углом

Связи «углерод–углерод» могут вращаться друг относительно друга по углу внутреннего вращения. Поэтому цепь полиэтилена в растворе или расплаве может принимать различные конформации, т. е.

проявлять гибкость. В кристаллической фазе полимера вращения связей не происходит.

Одной из основополагающих характеристик макромолекул полимеров является молекулярно-массовое распределение, описывающее соотношение количеств макромолекул различной молекулярной массы в данном образце [2]. Так, например, прочность полимеров с ростом массовой доли молекул с высокой молекулярной массой возрастает, но только до определенного предела.

2. Генератор псевдослучайных чисел

В качестве ГПСЧ предлагается использовать Вихрь Мерсенна – генератор псевдослучайной величины, разработанный в 1997 году японскими учёными Макото Мацумото и Такудзи Нисимура. Данный ГПСЧ основывается на свойствах простых чисел Мерсенна и обеспечивает быструю генерацию псевдослучайных чисел. Отличительная особенность генератора – большой период. Реализации данного алгоритма включены во множество программных средств, например в стандартную библиотеку языка C++ стандарта C++11 под именем std::mt19937. Для проверки применимости ГПСЧ к текущей задаче, а именно для проверки равномерности распределения, был сгенерирован тестовый файл с выборкой вещественных чисел от 0 до 1 размером в 1 000 000 значений. Гистограмма выборки представлена на рис. 2.

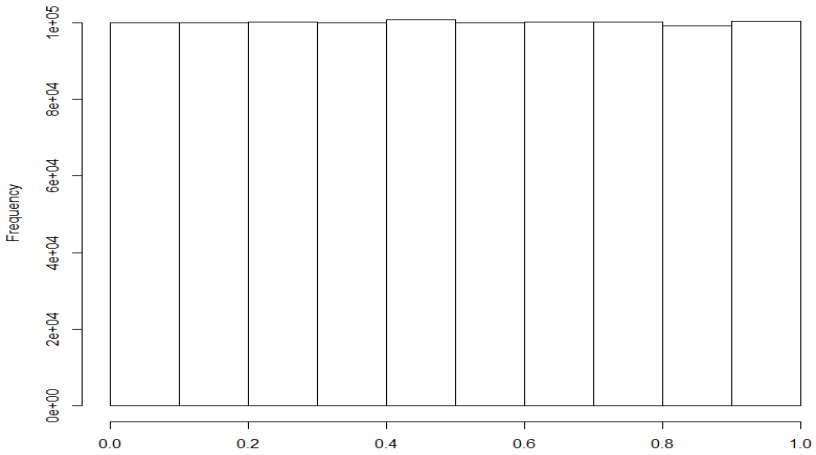


Рис. 2. Гистограмма случайных чисел по ГПСЧ «Вихрь Мерсенна»

Проверим выборку на равномерность распределения, используя критерий хи-квадрат.

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}. \quad (1)$$

здесь O_i – наблюдаемая частота интервала, а E_i – теоретическая. Примем количество интервалов равным 10, тогда степень свободы критерия хи-квадрат будет равна 9, а E_i будет одинаковой во всех интервалах и составит 100 000. В ходе расчетов было получено значение критерия 5.59772, что меньше критического значения 16.9 для 9 степени свободы с уровнем значимости 5%. Таким образом, нулевая гипотеза о равномерном распределении не отклоняется.

Для проверки по критерию Шапиро–Уилка W к полученной выборке случайных чисел было применено преобразование Бокса–Мюллера, позволяющее получить из равномерно распределенных значений величины, распределенные по нормальному закону. Далее были взяты первые 50 значений из выборки, так как данный критерий предназначен для выборок небольших размеров, и затем был вычислен непосредственно сам критерий по формуле

$$W = \frac{\left[\sum_{i=1}^{k} a_{n-i+1}(x_{n-i+1} - x_i)\right]^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}, \quad (2)$$

где a – табличные коэффициенты критерия, x_i, $1 \le i \le N$, – элементы выборки, $\overline{x}$ – математическое ожидание выборки. В ходе расчетов было получено значение критерия $W = 0.95797$, что выше критического значения для уровня значимости 5%, следовательно, нет оснований для отклонения нулевой гипотезы о нормальности распределения преобразованной выборки.

Таким образом, ГПСЧ «Вихрь Мерсенна» представляет собой качественный источник случайных чисел, подчиняющихся закону равномерного распределения.

3. Процедура построения единичной молекулы полимера

Существует несколько видов конформации молекул, например: свободно-сочлененная цепь и цепь с фиксированными валентными углами. При моделировании молекулы полимера в трехмерном евклидовом пространстве каждый мономер характеризуется тремя пространственными координатами, при этом процесс построения цепи

можно представить как последовательный процесс вычисления всех координат мономеров. Координаты первого мономера задаются конкретной точкой в пространстве или произвольным образом; для расчета координат второго мономера (а в случае свободно сочлененной цепи – для всех последующих) можно применить углы Эйлера, описывающие последовательную комбинацию пассивных поворотов вокруг осей вращающейся системы координат. Матричная форма углов Эйлера R представлена ниже:

$$\begin{pmatrix} \cos\alpha\cos\gamma-\cos\beta\sin\alpha\sin\gamma & -\cos\gamma\sin\alpha-\cos\alpha\cos\beta\sin\gamma & \sin\beta\sin\gamma \\ \cos\beta\cos\gamma\sin\alpha+\cos\alpha\sin\gamma & \cos\alpha\cos\beta\cos\gamma-\sin\alpha\sin\gamma & -\cos\gamma\sin\beta \\ \sin\alpha\sin\beta & \cos\alpha\sin\beta & \cos\beta \end{pmatrix}, \quad (3)$$

где α,β,γ – углы поворота вокруг координатных осей, которые определяются с помощью ГПСЧ согласно определенному закону распределения. Вычисление координат мономера в таком случае производится следующим образом: строится вектор, по норме совпадающий с длиной межмолекулярной связи (для простоты его можно совместить с одной из координатных осей) далее производится умножение R на вектор-столбец $\{x,y,z\}$, где x,y,z – координаты вектора. Чтобы получить конечный результат, полученные координаты складываются с соответствующими координатами предыдущего мономера.

Модель молекулы полимера со свободно-сочлененной цепью, построенная с помощью углов Эйлера согласно закону равномерного распределения и отображенная с помощью разработанного в среде Microsoft Visual Studio 2019 программного обеспечения, представлена на рис. 3.

В случае цепи с фиксированными валентными углами применяется матрица поворота вокруг произвольной оси

$$M(\bar{v},\theta)=\begin{pmatrix} \cos(\theta)+(1-\cos(\theta))x^2 & (1-\cos(\theta))xy-\sin(\theta)z & (1-\cos(\theta))xz+\sin(\theta)y \\ (1-\cos(\theta))yx+\sin(\theta)z & \cos(\theta)+(1-\cos(\theta))y^2 & (1-\cos(\theta))yz-\sin(\theta)x \\ (1-\cos(\theta))zx-\sin(\theta)y & (1-\cos(\theta))zy+\sin(\theta)x & \cos(\theta)+(1-\cos(\theta))z^2 \end{pmatrix}, \quad (4)$$

где $\bar{v}=(x,y,z)$ – единичный вектор, задающий ось вращения, а θ – угол поворота. В таком случае процесс нахождения координат каждого n_{i+1} мономера при известных n_i и n_{i-1} будет происходить в несколько этапов. Введем следующий вектор:

$$\bar{n}' = \bar{n}_{i-1} - \bar{n}_i . \quad (5)$$

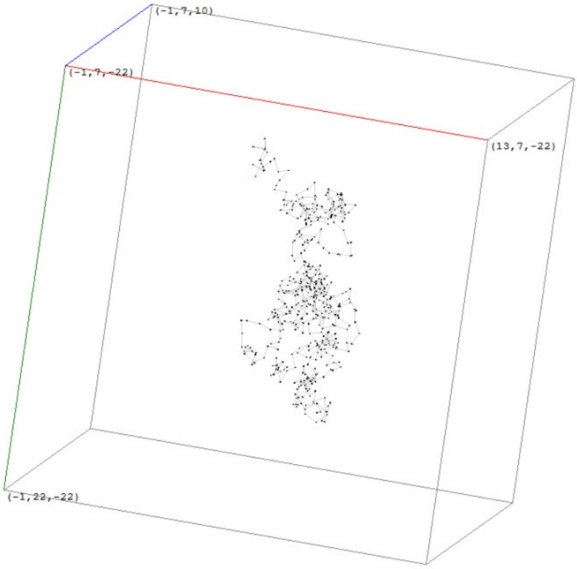


Рис. 3. Модель свободно-сочлененной молекулы

Данный вектор подвергается нормировке, чтобы избежать накопления ошибки в процессе работы компьютерной программы. Дополнительно введем точку n' с идентичными вектору $\overline{n}'$ координатами, тогда координаты n_{i+1} мономера можно получить путем двух поворотов точки относительно вектора в пространстве, сначала повернув точку n' вокруг нормали к вектору $\overline{n}'$ на угол $180^\circ - \alpha$, где α – заданный валентный угол, затем повернув еще раз, но уже вокруг самого вектора $\overline{n}'$ на угол β, $0^\circ \le \beta \le 360^\circ$, заданный случайно. В случае, если длина межмолекулярной связи не равна единице, производятся соответствующие вспомогательные преобразования. Вектор нормали $\overline{n}$ к вектору $\overline{v}$ можно найти путем фиксации двух из трех пространственных координат вектора $\overline{v}$, например x и y. Модель молекулы полимера с фиксированным валентным углом $\alpha = 75^\circ$ представлена на рис. 4.

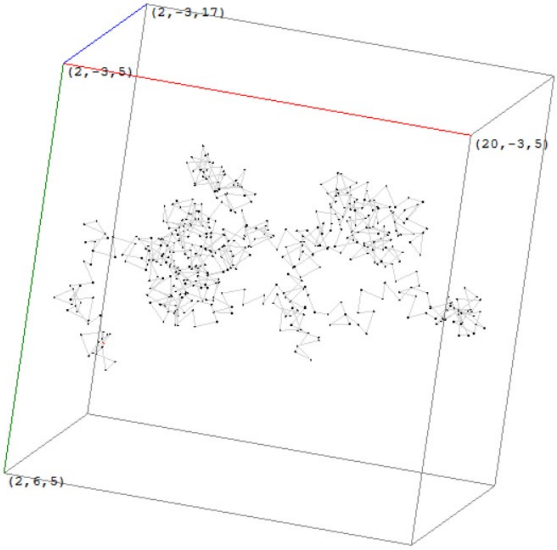


Рис. 4. Модель молекулы с фиксированным валентным углом

4. Фрактальная размерность

Для характеристики фрактальных структур, к коим можно отнести и мономеры в полимерной молекуле, принято использовать термин «фрактальная размерность» [3]. Данная характеристика является единственной измеримой для структур подобного рода. Используем размерность Минковского.

Процедура вычисления фрактальной размерности Минковского производится в несколько этапов. Сначала объект покрывается сеткой с ячейками известного размера ε, затем подсчитывается количество ячеек $N(\varepsilon)$, которые содержат в себе исследуемый объект. Далее размер ячеек уменьшается, и, соответственно, количество ячеек, содержащих объект, увеличивается. Процедура повторяется многократно. В общем виде формула расчета размерности Минковского D представлена ниже:

$$D = \frac{\log N(\varepsilon)}{\log \frac{1}{\varepsilon}}. \qquad (6)$$

Произведем расчет фрактальной размерности Минковского для полимерной цепи из 10^5 мономеров с фиксированным валентным углом $\alpha = 75^\circ$ при 100 итерациях алгоритма, при этом ограничим минимальный размер ячейки ε значением, на порядок превосходящим

длину связи между мономерами. Согласно методу вычисления данной размерности, её значение будет равно угловому коэффициенту линии регрессии, построенной на плоскости по рядам значений $\log N(\varepsilon)$ и $\log(1/\varepsilon)$.

Рассчитаем размерность для полимерной молекулы с углом внутреннего вращения, распределенным по нормальному закону с параметрами: среднеквадратичное отклонение $\sigma^2 = 1$, медиана $\mu = 0$. График размерности представлен на рис. 5. Значение размерности составило 1.8017.

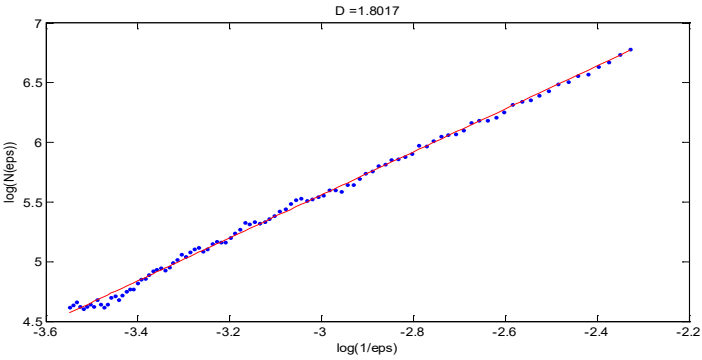


Рис. 5. Размерность Минковского для полимерной цепи с нормальным законом распределения для угла внутреннего вращения

Заключение

В качестве продолжения данной работы предлагается создание на основе данных алгоритмов и разработанного программного обеспечения программного комплекса на языке Python, позволяющего проводить имитационное моделирование радиолиза различных полимеров, обладающих большим многообразием конфигураций и конформаций, а также соотносить теоретические и экспериментальные данные. В комплексе планируется учитывать межмолекулярное взаимодействие полимерных цепей и его возможные вероятностные характеристики. Это позволит не только создавать полимерные материалы с определенными свойствами, но и утилизировать/перерабатывать уже готовые изделия, что может иметь, в том числе, и значительный экологический эффект.

Список литературы

1. Цветков, В. Н. Структура макромолекул в растворе / В. Н. Цветков, В. Е. Эскин, С. Я. Френкель. – М. : Наука, 1964. – 275 с.
2. Имитационное моделирование процесса деструкции полимера / Ю.К. Гусев [и др.] // Вестник ВГУИТ. – 2012. – №2. – С. 85-87.
3. Кузнецов, С. П. Динамический хаос : Курс лекций : Учеб. пособие для студентов вузов, обучающихся по физ. специальностям / С. П. Кузнецов. – М. : Физматлит, 2001. – 295 с.

Разработка программного обеспечения для создания цифровых водяных знаков в файлах контейнерах аудио формата

Е. Е. Ивашкина

Студент магистрант

Е. Ю. Митрофанова

Доцент

Введение

На данный момент интернет доступен практически в любой точке мира, что создает потрясающую и несуществующую раньше возможность обмениваться абсолютно любыми данными, находить их, выкладывать в общий доступ или скачивать. Данная статья посвящена встраиванию цифровых водяных знаков (далее – ЦВЗ) в файлы аудио формата.

1. Обоснование выбора аудио контейнера формата MP3

Формат файла определяет структуру и особенности представления звуковых данных при хранении на цифровом носителе.

В данной работе основное внимание уделяется файлам формата MP3.

Структура MP3 файла состоит из тегов в начале и конце файла, а также последовательности зависимых фрагментов (фреймов), которые в свою очередь имеют заголовок и блок данных (рис. 1).

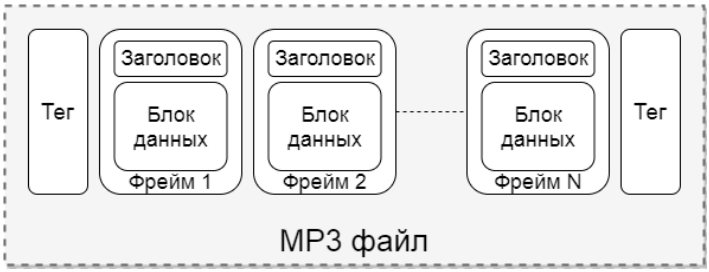


Рис. 1. Структура MP3 файла

2. Классификация методов встраивания ЦВЗ

На данный момент существует большое количество методов, которые в основном используются для встраивания водяных знаков в изображения, однако некоторые из этих методов могут быть применимы и для контейнеров аудиоформата.

Реализуемый метод должен соответствовать следующим основным требованиям [1]:

1. Прозрачность;
2. Устойчивость;
3. Пропускная способность.

Метод замены наименее значащего бита (Least Significant Bits, LSB). Использование метода LSB базируется на невосприимчивости человеческих органов чувств к малозначительным изменениям в контейнерах, обладающих психовизуальной избыточностью. Его реализация заключается в замене последних значащих бит в контейнере на биты сообщения (водяного знака), причем разница между пустым контейнером и контейнером с сообщением не ощутима для восприятия человеком [2].

Основными достоинствами метода можно назвать его высокую пропускную способность, прозрачность, легкость реализации и универсальность – он подходит для контейнеров любого формата. К недостаткам относится его неустойчивость – сообщение разрушается практически при любых воздействиях на контейнер.

Метод фазового кодирования. Метод фазового кодирования предполагает замену фазу исходного аудио сегмента на опорную фазу, которая представляет сообщение для скрытия [3]. Алгоритм фазового кодирования подразумевает следующие шаги:

1. Разделение оригинального аудио сигнала $S[i], (1 \le i \le I)$ на серию N коротких сегментов $S_n[i], (1 \le n \le N)$;

2. Применение k-точечного ДПФ к n-му сегменту сигнала $s_n[i]$, где $K = I / N$ и создание матрицы фаз $\phi_k(w_k)$ и амплитуд $A_k(w_k)$ для $(1 \le k \le K)$;

3. Вычисление разности фаз между соседними сегментами $\Delta\phi_k(w_k)$;

4. Так как фазовые сдвиги могут быть легко обнаружены, в стегосигнале должны быть сохранены разности фаз. В связи с этим, сообщение встраивается только в фазу первого сегмента в соответствии с формулой 1 ниже:

$$\text{Новая Фаза} = \begin{cases} \frac{\pi}{2}, \text{бит Сообщения} = 0 \\ -\frac{\pi}{2}, \text{бит Сообщения} = 1 \end{cases} \quad (1)$$

5. Используя новую фазу первого сегмента создаются новая матрица фаз и разности между ними.

Звуковой сигнал восстанавливается с использованием обратного дискретного преобразования Фурье.

Достоинством данного метода является его безопасность (для извлечения сообщения необходимо знать длину аудио сегмента), прозрачность. К недостаткам относится достаточно низкая пропускная способность и сложность реализации метода.

3. Реализация программного средства

Используя описанные выше методы, разработанное приложение позволяет встраивать ЦВЗ и извлекать его, включая поддержку проигрывания сигнала и отображения дополнительной информации, как о самом сигнале, так и о результате встраивания. В качестве основного языка программирования был выбран C# из-за кроссплатформенности, наличия большого количества бесплатных дополнительных библиотек и шаблонов, а также из-за удобства работы с графическим интерфейсом. Главными используемыми библиотеками являются NAudio [4] и COM-компонента WMPLib.

На рис. 2 представлена диаграмма вариантов использования, которая иллюстрирует доступные возможности пользователя при работе с приложением. Помимо целевых действий («Извлечь сообщение», «Создать ЦВЗ», включающие несколько других зависимых пунктов), приложение имеет дополнительно опции анализа результата.

Для встраивания ЦВЗ необходимо выбрать MP3 файл при нажатии на кнопку «Открыть исходный файл», ввести сообщение для скрытия в

текстовом поле, выбрать метод, который будет использоваться для встраивания ЦВЗ и нажать кнопку «Провести встраивание». В случае, если какой-либо информации недостаточно (например, пользователь не указал аудио файл), программа отобразит информационное сообщение с подсказкой. При успешном встраивании, заполненный контейнер замещает собой исходный в плеере и аудиозапись можно прослушать или сохранить. Для извлечения ЦВЗ достаточно только указать контейнер, выбрать метод и нажать кнопку «Извлечь». Результат отобразится в текстовом поле на одноименной вкладке (рис. 3).

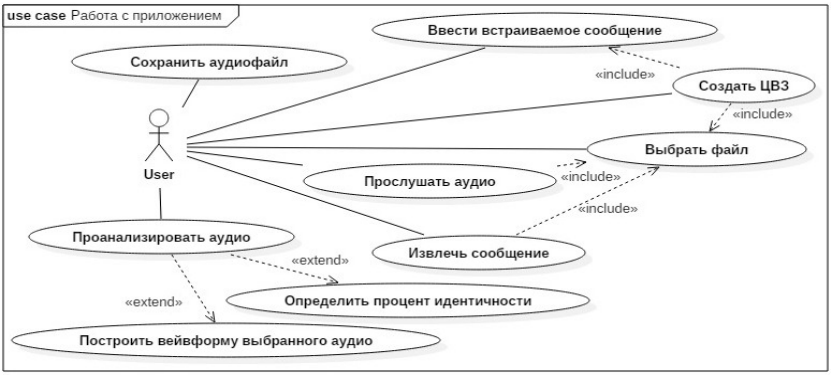


Рис. 2. Диаграмма вариантов использования

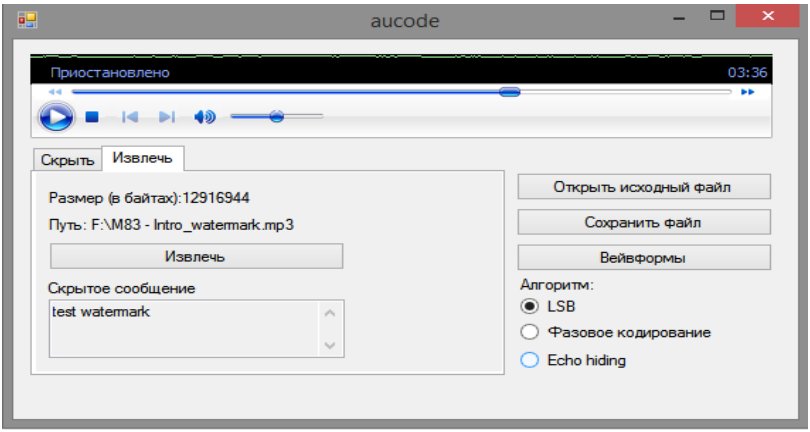


Рис. 3. Извлечение ЦВЗ

Для пользователя предусмотрены следующие функциональные возможности:

– открыть исходный или конечный аудиофайл (контейнер) и прослушать его, используя встроенный плеер;

– получить информацию о размере аудиофайла в байтах;

– скрыть или извлечь сообщение;

– получить больше информации об используемом методе в виде подсказки (рис. 4).

При нажатии на кнопку «Проанализировать аудио» открывается новое окно (рис. 5) с вейвформами для исходного и полученного аудио. В верхней части окна располагается панель меню, включающая набор опций для отображения каждого файла, список анализируемых параметров (например, «Найти» → «Идентичность» − рис. 6), а также имена сравниваемых файлов.

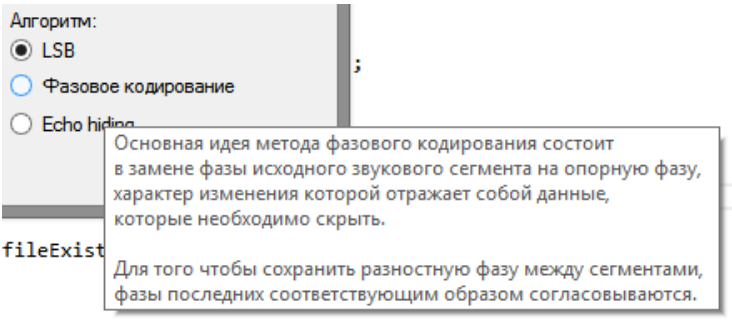


Рис. 4. Подсказка при наведении на название метода

Рис. 5. Вейвформы аудиозаписей до (вверху) и после (внизу) встраивания ЦВЗ с демонстрацией опций для одного из файлов

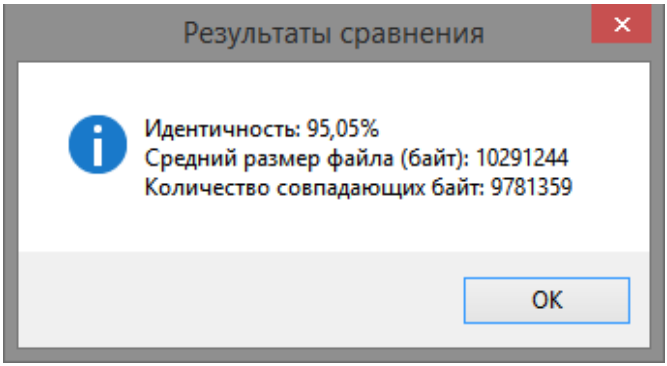


Рис. 6. Сообщение о результатах сравнения идентичности для выбранных аудиофайлов

Заключение

Разработанное программное обеспечение позволяет реализовывать встраивание и извлечение информации в файлах контейнерах аудио форматов, а также проводить анализ полученных результатов.

Список литературы

1. Bii, J. K. MPEG-1 Layer III Standard: A Simplified Theoretical Review / J. K. Bii // International Journal of Advanced Research in Computer Science and Electronics Engineering. – 2019. – № 8. – P. 45-49.

2. Стеганография звука методом LSB [Электронный ресурс] : Национальная библиотеки им. Н. Э. Баумана. – Режим доступа : https://ru.bmstu.wiki/Стеганография_звука_методом_LSB

3. Atoum, M. S. MP3 Steganography: Review / M. S. Atoum, G. Sulong // International Journal of Computer Science Issues. – 2012. – № 9. – P. 236-244.

4. Audio and MIDI library for .NET [Электронный ресурс] : репозиторий. – Режим доступа : https://github.com/naudio/NAudio

Моделирование поведения балки под нагрузкой

Е. А. Карпов

Студент магистрант

М. Е. Семенов

Профессор

Введение

В настоящее время проектирование строительных конструкций является неотъемлемой частью строительного процесса. При этом первым этапом является моделирование поведения строительных конструкций под воздействием различных внешних воздействий. Результаты математического моделирования позволяют конструировать сейсмоустойчивые и аварийно-устойчивые конструкции. Известно, что одним из базовых и важнейших элементов, в любых несущих конструкциях, является балка. Настоящая работа посвящена исследованию динамических режимов балки, находящейся под воздействием внешнего периодического воздействия.

Математическая модель колебаний балки обычно описывается посредством теории Эйлера–Бернулли [1], или же посредством более поздней теории Тимошенко [2]. Основное различие между указанными подходами заключается в том, что в теории Эйлера–Бернулли предполагается, что поперечное сечение всегда перпендикулярно нейтральной оси, несмотря на деформацию, тогда как теория Тимошенко является обобщением теории Эйлера-Бернулли: в ней предполагается, что при деформации балки поперечное сечение не перпендикулярно нейтральной оси и напрямую зависит от деформации. В настоящей работе для описания колебаний балки используется теория Эйлера–Бернулли:

$$u_{tt} + a^2 u_{xxxx} = g(x,t), \quad 0 < x < L, \quad t > 0, \tag{1}$$

где нижние индексы характеризуют частные производные, L – длина балки, $a = \text{const}$ – характеризует физические свойства балки (модуль упругости, плотность и т. д), $g(x,t)$ – управляющее воздействие, $u(x,t)$ – отклонение от положения равновесия. Начальные и граничные условия формализуются следующим образом:

$$u(x,0) = u_t(x,0) = 0, \tag{2}$$

$$u(0,t) = u_{xx}(0,t) = u(L,t) = u_{xx}(L,t) = 0,$$

граничные условия соответствуют шарнирному закреплению балки в концевых точках, а начальные условия соответствуют покоящейся балке. Большинство последних работ по исследованию балок посвящены анализу собственных частот балок разного типа: «коробчатых» балок [3] и изогнутых балок [4].

При проектировании строительных сооружений наиболее информативной характеристикой является реакция сооружения на внешнее воздействие с определенной частотой. Таким образом, интересна реакция балки на периодическое внешнее возмущение на определенных частотах. Это характеристика называется АЧХ (амплитудно-частотная характеристика). Построение АЧХ является главной задачей настоящей работы. Для этого понадобится реализация разностной схемы, описание которой также приводится в настоящей работе.

1. Разностная схема в явном виде

В теории численных методов подход, преобразующий дифференциальное уравнение (в общем случае систему дифференциальных уравнений) в частных производных к конечной системе алгебраических уравнений, называется разностной схемой. Этот подход применяется, прежде всего, в случаях, когда аналитическое решение невозможно получить, либо оно неизвестно. На практике наиболее часто используется следующая классификация разностных схем: явные и неявные схемы. Неявная схема предполагает решение СЛУ (системы линейных уравнений) на каждом временном слое, тогда как в явной схеме результат сетки на $(n+1)$-м шаге вычисляется по уже известным результатам на предыдущих шагах. В настоящей работе используется явная разностная схема.

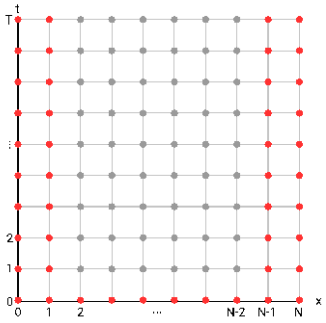
Введем в области $\Omega = \{0 \le x \le L, 0 \le t \le T\}$ равномерную сетку с шагом $h_t = T / M$ (M – количество узлов в сетке по временной координате, T – время моделирования) по времени и $h_x = L / N$ (N – количество узлов в сетке по пространственной координате) по пространству (см. рис. 1). Вторые производные по времени и пространству и четвертую производную по пространству аппроксимируем следующим образом:

$$u_{xx} \to \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h_x^2}, \qquad (3)$$

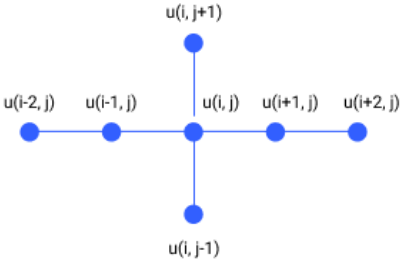
$$u_{tt} \to \frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{h_t^2}, \tag{4}$$

$$u_{xx} \to \frac{6u_i^j - 4u_{i-1}^j - 4u_{i+1}^j + u_{i+2}^j + u_{i-2}^j}{h_x^4}, \tag{5}$$

где нижние и верхние индексы (i и j) характеризуют пространственный и временной слои, соответственно.



а



б

серым отмечены неизвестные значения в узлах, красным – известные

Рис. 1. Область разбиения: а – сетка явной разностной схемы, б – шаблон явной схемы уравнения колебаний балки

Отметим, что для того, чтобы явная схема не расходилась на начальном временном интервале, принято выбирать $h_t >> h_x$. Аппроксимация граничных и начальных условий имеет следующий вид:

$$u(0,t) = u(L,t) = 0 \to u_0^j = u_L^j = 0,$$
$$u_{xx}(0,t) = u_{xx}(L,t) = 0 \to u_1^j = \frac{u_2^j + u_0^j}{2}, u_{N-1}^j = \frac{u_{N-2}^j + u_N^j}{2}, \tag{6}$$

$$u(x,0) = u_t(x,0) = 0 \to u_0^j = u_1^j = 0. \tag{7}$$

Теперь, когда аппроксимация производных совместно с граничными и начальными условиями известна, подставляя (3) – (7) в (1), (2), получим:

$$u_i^{j+1} = 2u_i^j - u_i^{j-1} - h_t a^2 \frac{6u_i^j - 4u_{i-1}^j - 4u_{i+1}^j + u_{i+2}^j + u_{i-2}^j}{h_x^4} + h_t g(ih_x, jh_t). \quad (8)$$

2. Результаты

В настоящей работе внешнее воздействие задается в виде периодического сигнала с меняющейся частотой. Под АЧХ будем понимать зависимость отклонения положения балки от положения равновесия в фиксированной точке пространства от частоты внешнего периодического сигнала. Амплитуда колебаний балки вычислялась следующим образом: численно решалось уравнение (1) при помощи явной разностной схемы (8), фиксировалась пространственная координата, вследствие чего получалась зависимость колебаний балки в фиксированной точке от времени. У полученной зависимости искался максимум и минимум, после чего вычислялась абсолютная разность максимума и минимума, деленная на два. Полученную АЧХ для балки с параметрами $L = 1$, $T = 5$, с шагом сетки по пространственной координате, равным 0.05, по временной координате равным 0.00125, $a^2 = 4/5$, можно видеть на рис. 2, а результаты численного решения уравнения (1) с функциями нагрузки, равными $2\sin(t)$ и $5\sin(2t)$, можно видеть на рис. 3. (а) и рис. 3. (б), соответственно.

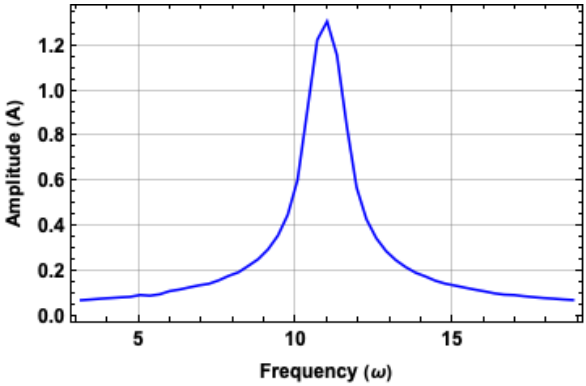
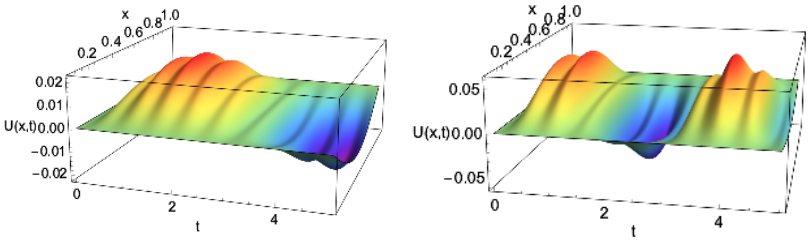


Рис. 2. Амплитудно-частотная характеристика колебаний балки при нагрузке $g(x.t) = 5\sin(\omega t),\ \omega = \pi..20\pi$



а б

параметры $L = 1.T = 5, a^2 = 4/5, h_x = 0.05, h_x = 0.00125$

Рис. 3. График численного решения уравнения (1):
а – $g(x.t) = 2\sin(t)$, б – $g(x.t) = 5\sin(2t)$

Таким образом, в рамках настоящей работы была построена амплитудно-частотная характеристика, с помощью которой была идентифицирована собственная частота колебаний балки, равная примерно 10.98.

3. Обсуждение

Отметим, что одно из перспективных направлений, связанных с моделированием колебаний балки, относится к моделированию материалов, обладающих гистерезисной природой. В связи с этим уравнение описания динамики балки значительно усложняется. Кроме того, одним из свойств гистерезисных преобразователей является диссипация энергии [5]. Учет гистерезиса позволит более точно моделировать реакцию балки на внешнее воздействие, а также строить более безопасные конструкции.

Заключение

Настоящая статья посвящена исследованию динамических режимов балки, описываемой уравнениями Эйлера–Бернулли, под воздействием внешнего периодического сигнала. В частности, представлена явная разностная схема, аппроксимирующая уравнения колебаний балки. Приведены результаты решения уравнения при различном внешнем воздействии, а также получена зависимость амплитуды колебаний балки от частоты внешнего воздействия (амплитудно-частотная характеристика), формализуемого посредством периодического сигнала с меняющейся частотой. В дальнейшем предполагается разработать математический аппарат, описывающий

колебания балки, в каждом узле которой находится гистерезисный преобразователь.

Список литературы

2. Seon, M. H. Dynamics of transversely vibrating beams using four engineering theories / M. H. Seon, B. Haym, W. Timothy // Journal of Sound and Vibration. – 1999. – Vol. 225. – Issue 5. – P. 935-988.

3. Timoshenko, S. P. History of strength of materials / S. P. Timoshenko. – New York : Courier Corporation, 1983. – 452 p.

4. Choi, S. Consistent higher-order beam theory for thin-walled box beams using recursive analysis: Edge-bending deformation under doubly symmetric loads / S. Choi, Y. Y. Kim // Engineering Structures. – 2020. – Vol. 206. – Article 110129.

5. Atteshamuddin, S. S. A sinusoidal beam theory for functionally graded sandwich curved beams / S. S. Atteshamuddin, M. G. Yuwaraj // Composite Structures. – 2019. – Vol. 226. – Article 111246.

6. Bouc-Wen model of hysteretic damping / A. M. Solovyov [et al.] // Procedia Engineering. – 2017. – Vol. 201. – P. 549-555.

Настоящая работа выполнена при поддержке гранта РФФИ №19-08-00158-а.

Исследование систем билинейных уравнений с помощью символьных вычислений

Д. В. Козориз

Студент магистрант

А. В. Лобода

Профессор

Введение

В задаче описания однородных поверхностей многомерных комплексных пространств возникают системы уравнений, зависящих линейным образом от двух групп переменных и имеющих достаточно большие размеры.

В частности, для рассматриваемых в настоящее время вещественных гиперповерхностей 4-мерного комплексного пространства одна из таких групп переменных содержит 27

неизвестных величин, а другая – 9. Количество уравнений в обсуждаемой «однородной» (имеющей нулевую правую часть) системе равно размерности пространства первой группы, т. е. 27.

1. Постановка задачи

В 27-мерном пространстве вещественных многочленов $N_{22}(z_1, z_2, z_3)$ от трех комплексных переменных, удовлетворяющих т. н. tr-условию Мозера [1], изучается задача «сохранения» отдельного элемента подгруппой унитарной группы $U(3)$, т. е.:

$$N_{220}(U(t)z, U(t)\overline{z}) = N_{220}(z, \overline{z}), \tag{1}$$

$$\langle U(t)z, U(t)z \rangle = \langle z, z \rangle. \tag{2}$$

Здесь $\langle z, z \rangle = |z_1|^2 + |z_2|^2 + |z_3|^2$ – эрмитова форма, $U(t)$ – унитарная матрица 3-го порядка, зависящая от параметра t.

Отдельных вычислений требует построение базиса пространства обсуждаемых многочленов, являющегося подпространством пространства однородных многочленов типа (2, 2) и имеющего вещественную размерность 36.

Эту задачу удобнее рассматривать после перехода от группы Ли к ее алгебре Ли за счет дифференцирования равенства (1) по параметру t. Матрица $X = U'(0)$ является косоэрмитовой, содержит 9 вещественных параметров и имеет вид:

$$X = \begin{pmatrix} i\alpha_1 & \beta_1 + i\beta_2 & \beta_3 + i\beta_4 \\ -\beta_1 + i\beta_2 & i\alpha_2 & \beta_5 + i\beta_6 \\ -\beta_3 + i\beta_4 & -\beta_5 + i\beta_6 & i\alpha_3 \end{pmatrix} \tag{3}$$

Учитывая, что $U(0) = E$, получим из (1) систему 27 уравнений, линейных относительно 27 коэффициентов многочлена N_{220} и 9 элементов матрицы X.

Полученную систему билинейных уравнений можно представить как линейную (27×27)-систему относительно координат $(\alpha_1, \alpha_2, \alpha_3, \beta_1, ..., \beta_6)$ или (27×9)-систему относительно коэффициентов $(\lambda_1, ..., \lambda_{27})$ многочлена в N_{22}.

Ниже в качестве примера представлены первые 7 строк первых пяти столбцов (27×9)-матрицы:

$$M_2 = \begin{pmatrix} 0 & 0 & 0 & -\lambda_4 + \lambda_6 & -\lambda_5 - \lambda_7 \\ 0 & 0 & 0 & -\lambda_4 - \lambda_6 & -\lambda_5 + \lambda_7 \\ 0 & 0 & 0 & \lambda_4 + \lambda_6 & \lambda_5 - \lambda_7 \\ \lambda_5 & -\lambda_5 & 0 & 6\lambda_1 + 2\lambda_2 - 2\lambda_{16} & 2\lambda_{17} \\ -\lambda_4 & \lambda_4 & 0 & 2\lambda_{17} & 6\lambda_1 + 2\lambda_2 + 2\lambda_{16} \\ -\lambda_7 & \lambda_7 & 0 & -6\lambda_1 - 2\lambda_3 + 2\lambda_{16} & -2\lambda_{17} \\ \lambda_6 & -\lambda_6 & 0 & 2\lambda_{17} & 6\lambda_1 + 2\lambda_3 + 2\lambda_{16} \end{pmatrix} \quad (4)$$

Отметим, что характерной особенностью полученной матрицы является её «сплошная» заполненность (за исключением первых трёх столбцов).

Например, для многочлена $E_1 = |z_1|^4 - 4|z_1|^2|z_2|^2 + |z_2|^2$ (один из базисных элементов пространства многочленов) матрица M_2 будет иметь вид:

$$\begin{pmatrix} 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2. Исследование ранга матрицы

Теоретические рассмотрения работы [2] позволяют утверждать, что в данной постановке при любом ненулевом векторе переменных из первого пространства ранг (27×9)-матрицы оказывается не меньше 4.

Одной из целей настоящей работы является проверка этого утверждения. Второй целью является выявление всех возможных векторов из первого пространства, обеспечивающих именно такой минимальный ранг изучаемой матрицы. Тем самым, предлагается

очертить потенциально возможное множество «максимально симметричных» однородных поверхностей в терминах тейлоровских коэффициентов их определяющих функций.

Отметим, что в аналогичной задаче в 3-мерном комплексном пространстве [3] соответствующая матрица билинейной системы имеет размеры (5×4), и вычисления всех необходимых миноров этой матрицы удалось реализовать [3] фактически в ручном режиме. Однако количества миноров, например, 4-го и 5-го порядков у (27×9)-матрицы составляют, соответственно, $C_{27}^4 \cdot C_9^4 = 2211300$ и $C_{27}^5 \cdot C_9^5 = 10171980$, а потому привлечение информационно-компьютерных технологий является в этой задаче вполне естественным.

Напомним, что необходимым условием равенства ранга изучаемой матрицы числу 4 является обращение в нуль всех миноров 5-го порядка этой матрицы. Метод решения поставленной задачи заключается в выделении относительно небольшого количества «удобных» миноров 5-го порядка, зависящих от векторов из первого пространства.

Назовем «удобными» миноры следующего вида:

$$c_1 \cdot \lambda_i \cdot \lambda_j \cdot \lambda_g (c_2 \lambda_k \lambda_l + c_3 \lambda_p \lambda_q), \tag{5}$$

где c_1, c_2, c_3 – числовые коэффициенты, отличные от нуля.

Таковых в начальном пакете из 208272 ненулевых миноров пятого порядка оказалось 1321. Составим из них систему и решим её (решение в Maple занимает несколько секунд). Получим 15 решений, поочередно подставим их и рассмотрим уже более разреженные матрицы.

Например, на рисунке представлен вид матрицы после подстановки первого решения.

Изучая получившуюся матрицу, можно выделить миноры, позволяющие обнулить ещё некоторые коэффициенты:

$$\begin{cases} -32\lambda_{20}(\lambda_{20}^2 + \lambda_{21}^2)^2 = 0 \\ 32\lambda_{21}(\lambda_{20}^2 + \lambda_{21}^2)^2 = 0 \\ \lambda_{27}(\lambda_{26}^2 + \lambda_{27}^2)^2 = 0 \\ -\lambda_{26}(\lambda_{26}^2 + \lambda_{27}^2)^2 = 0 \\ -4\lambda_6^2(-\lambda_6 + \lambda_{12})^2(\lambda_6 - \lambda_{12}) = 0 \\ -32\lambda_{12}^3(\lambda_6 - 5\lambda_{12})^2(-\lambda_6 + 5\lambda_{12}) = 0 \end{cases}$$

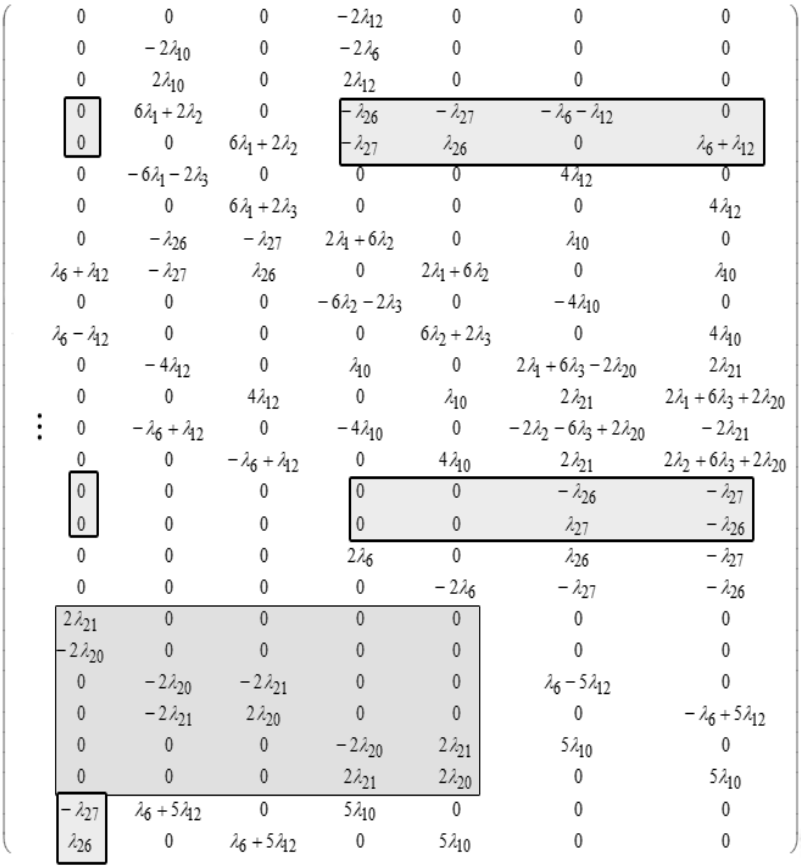


Рисунок. Вид матрицы после подстановки решения с выделенными подматрицами

Наконец, после обнуления λ_6, λ_{12}, λ_{20}, λ_{21}, λ_{26}, λ_{27} можно найти минор $-1250\lambda_{10}^5$, а значит и $\lambda_{10} = 0$. Теперь матрица имеет вид (представлены не все строки и столбцы):

$$\begin{pmatrix} 6\lambda_1+2\lambda_2 & 0 & 0 & 0 \\ 0 & 6\lambda_1+2\lambda_2 & 0 & 0 \\ -6\lambda_1-2\lambda_3 & 0 & 0 & 0 \\ 0 & 6\lambda_1+2\lambda_3 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2\lambda_1+6\lambda_3 & 0 \\ 0 & 0 & 0 & 2\lambda_1+6\lambda_3 \\ 0 & 0 & -2\lambda_2-6\lambda_3 & 0 \\ 0 & 0 & 0 & 2\lambda_2+6\lambda_3 \end{pmatrix} \quad (6)$$

В (6) содержится 193 ненулевых минора, уникальных из них – 54. Составим и решим систему (для этого удобно воспользоваться методом базисов Грёбнера [4]):

$$\begin{cases} (3\lambda_1+\lambda_2)^2(\lambda_1+3\lambda_2)^2(\lambda_2+3\lambda_3)=0 \\ (3\lambda_1+\lambda_2)^2(\lambda_1+3\lambda_2)(3\lambda_2+\lambda_3)(\lambda_1+3\lambda_3)=0 \\ \ldots \\ (3\lambda_1+\lambda_3)(3\lambda_2+\lambda_3)^2(\lambda_2+3\lambda_3)^2=0 \end{cases} \quad (7)$$

Помимо тривиального, получим 3 решения:

$$\{\lambda_1=-\frac{1}{3}\lambda_3, \lambda_2=\lambda_3, \lambda_3=\lambda_3\}, \{\lambda_1=\lambda_3, \lambda_2=-\frac{1}{3}\lambda_3, \lambda_3=\lambda_3\},$$
$$\{\lambda_1=-3\lambda_3, \lambda_2=-3\lambda_3, \lambda_3=\lambda_3\}.$$

Таким образом, минимальный ранг матрицы M_2 равен 4. Для многочлена $N_1=-3E_1-3E_2+E_3$, матрица выглядит так (ненулевой фрагмент):

$$\begin{pmatrix} -24 & 0 & 0 & 0 \\ 0 & -24 & 0 & 0 \\ 16 & 0 & 0 & 0 \\ 0 & -16 & 0 & 0 \\ 0 & 0 & -24 & 0 \\ 0 & 0 & 0 & -24 \\ 0 & 0 & 16 & 0 \\ 0 & 0 & 0 & -16 \end{pmatrix}$$

Заключение

Разработана техническая идея определения рангов больших матриц, линейно зависящих от многомерного параметра. Идея апробирована на малых рангах с привлечением компьютерных алгоритмов построения базисов Грёбнера систем полиномиальных уравнений. Также был определен минимальный ранг матрицы M_2 и найден пример многочлена, обеспечивающий его.

Данная постановка возникла в задаче многомерного комплексного анализа; в то же время системы билинейных уравнений и определение их рангов имеют приложения в задачах экономики, социологии и др.

Список литературы

1. Chern, S. S. Real Hypersurfaces in complex manifolds / S. S. Chern, J. Moser // Acta Mathematica. – 1974. – № 133. – P. 219-271.

2. Kruglikov, B. Submaximally symmetric CR-structures / B. Kruglikov // Journal of Geometric Analysis. – 2015. – № 26. – P. 3090-3097.

3. Лобода, А. В. О размерности группы, транзитивно действующей на гиперповерхности в C^3 / А. В. Лобода // Функц. анализ и его прил. – 1999. – Т. 33. – Выпуск 1. – С. 68-71.

4. Аржанцев, И. В. Базисы Грёбнера и системы алгебраических уравнений / И. В. Аржанцев. – М. : МЦНМО, 2003. – 68 с.

Классификация трафика на основе машинного обучения

К. А. Коловатов

Студент магистрант

А. С. Коваль

Старший преподаватель

Введение

Сегодня тема классификации и фильтрации трафика довольно актуальна. Многие компании активно используют межсетевые экраны в своих дата-центрах для управления трафиком, какой трафик можно пропускать, а какой нежелательно. Допустим не желаем пропускать Torrent трафик, но как мы знаем, у данного протокола нет точно определенного порта, поэтому нам придется просматривать все пакеты и определять какой именно трафик является Torrent трафиком, а какой нет, данная работа поможет с легкостью классифицировать такой трафик и ограничить его.

Операторы связи классифицируют трафик для решения различных задач: информационной безопасности, обеспечения качества обслуживания, управление трафиком и перегрузками каналов. Сейчас тема классификации трафика стала особенно актуальной в связи с требованиями регулятора по запрету определенного содержимого или сайтов и увеличением объемов передаваемого трафика, что в свою очередь, предъявляет высокие требования к оборудованию и методам, которыми классифицируют трафик. Остаются и текущие, регулярные задачи, например, идентификация Torrent E-mule и подобных, иногда нежелательных для операторов связи, трафиков. В последнем случае, трафик невозможно идентифицировать по идентификатору в заголовке пакета, т.к. известная пятерка: порты и адреса источника, назначения и номер транспортного протокола, могут принимать любые значения. Данная работа ставит целью идентифицировать такой трафик.

1. Традиционные средства классификации

В настоящее время часто используется классический метод классификации трафика и его блокировку. Например, на основе порта приложения, который закреплен за ним [1].

Плюсы метода:

– Скорость.

– Не требовательно к ресурсам.

– Поддерживается многими устройствами

– Не нужна полезная нагрузка на уровне приложений, так что это не ставит под угрозу конфиденциальность пользователей компьютера.

Минусы метода:

– Легко обмануть, изменив номер порта.

– Полезно только для приложений и служб, которые используют номера фиксированного порта

Также есть технология DPI (Deep Packet Inspection), но данной технологии нужны фактическая полезная нагрузка (содержимое пакета) и статистические данные [2].

Плюсы метода:

– Обнаруживает приложения и услуги, независимо от номера порта, на котором они работают.

Минусы метода:

– Исследует фактическую полезную нагрузку пакета.

– Отсутствие поддержки для многих приложений, как Skype, который плохо поддерживается большинство классификаторов.

– Медленная.

– Требуется много вычислительной мощности.

– Данные должны быть всегда свежие, так как приложения меняются очень часто.

– Шифрование позволяет во многих случаях этот метод обнаружения сделать не рабочим.

Именно из-за минусов вышеперечисленных методов, в данной работе будет рассматриваться классификация на основе машинного обучения, дабы исключить возможность легкого обмана системы как в случае стандартных пакетных фильтров, так и в случае DPI, где главная проблема технологии в том, что она исследует фактическую полезную нагрузку пакета.

2. Среда разработки

Для реализации алгоритма был выбран язык Python. Этот язык ориентирован на повышение производительности разработчика и читаемости кода, имеет простой синтаксис и существуют отличные среды для разработки, такие как PyCharm от компании JetBrains.

Использовались следующие библиотеки Python:

1. Scikit-learn – самый распространенный выбор для решения задач классического машинного обучения.
2. Numpy – для поддержки больших многомерных массивов и матриц.
3. Pandas – обработка и анализ данных.

В данной работе был выбран алгоритм Random Forest (Случайное дерево), так как один из его плюсов это: способность эффективно обрабатывать данные с большим числом признаков и классов, а также существуют методы оценивания значимости отдельных признаков в модели [3].

3. Как использовалось машинное обучение

В Deep Packet Inspection, поток транспортного уровня является объектом классификации. Данная работа основывается на таких же потоках.

Приложения образовывают потоки транспортного уровня с уникальными статистическими характеристиками несмотря на то, что они пользуются даже разными протоколами.

Для того чтобы определить с максимальной точностью какой из приложений создает поток и понять какой протокол транспортного уровня в нем используется, нам потребуется правильно определить набор меток.

Характеристики потоков в данной работе будут определены около четырёх значений:

1. Очередь размеров секций транспортного уровня, отосланных от сервера.
2. Очередь размеров секций транспортного уровня, отосланных от клиента.
3. Очередь размеров доли данных, отосланных от сервера.
4. Очередь размеров доли данных, отосланных от клиента.

Вышеперечисленные четыре значения очень хорошо определяют поток данных, а уже на основе этих данных можно узнать протокол прикладного уровня.

Опираясь на эти 4 значения мною было составлено около 30 характеристик.

Несколько примеров характеристик:

- Протокол.
- Средний размер пакета от клиента.
- Средний размер пакета от сервера.

Так как важность этих характеристик определить пока невозможно, «Случайный Лес» как раз тот алгоритм, который поможет с этим, так как он мало чувствителен к шумам и корреляции признаков.

Выбранный алгоритм из разряда «обучение с учителем». Для этого разделим выборку на две, по первой выборке с заранее известными метками мы будем обучать нашу модель, а по второй производить проверку и оценивать на сколько хорошо алгоритм работает.

4. Тестовый стенд

Для исследования необходим перехваченный трафик, пакеты которого в дальнейшем поделим на обучающую и проверочную выборки. Для этого установили программу для прослушивания сети и записи всех пакетов, которые были переданы на трех компьютерах. А именно записывался трафик нескольких компьютеров, один из которых был подключен к роутеру через Wi-Fi, остальные два, кабелем.

5. Тестирование разработанного программного обеспечения

Берем поток данных, который был собран и преобразуем его в таблицу признаков в формате CSV с помощью модуля программы. Далее запускам машинное обучение, для этого реализован модуль, осуществляющий обучение модели на нашей таблице. После выполнения программы получим данные, представленные в табл. 1.

Таблица 1

Протоколы и количество собранных потоков по ним

№ п/п	Протокол	Количество потоков
1	HTTPS	7213
2	DNS	2501
3	HTTP	1871
4	Torrent	108

Теперь полученную выборку потребуется разделить на обучающую и проверочную, для того чтобы можно было обучить нашу модель. После того как это сделаем, мы получим количество потоков в обучающей выборе и в проверочной, данные представлены в табл. 2.

Таблица 2

Количество потоков в выборках

№ п/п	Протокол	Обучающая	Проверочная
1	HTTPS	2340	4843
2	DNS	1023	1478
3	HTTP	821	1050
4	Torrent	40	68
	Всего:	4224	7439

Теперь можно запускать ядро программы, которое обучит нашу модель и на основе обучающей выборки предоставит таблицу, в которой будет показаны значения точность, полнота, F-мера, количество

обучающей выборки по каждому протоколу. Данные представлены в табл. 3.

Таблица 3

Точность и полнота по классам

№ п/п	Протокол	Точность	Полнота	Ф-мера	Количество
1	HTTPS	1	0.99	1	4843
2	DNS	1	1	1	1478
3	HTTP	1	0.99	1	1050
4	Torrent	0.98	0.9	0.94	68

Также в табл. 4 в каждой ее ячейке число показывает количество моментов, когда трафик, определился правильно, т.е. слева представлены реальные классы, сверху предсказанные. Тем самым значения по диагонали показывают нам какие потоки были правильно классифицированы, а какие нет, т.е. вне диагонали это ошибки.

Таблица 4

Реальные и предсказанные классы

	№ п/п		Предсказанные классы			
Реальные классы			HTTPS	DNS	HTTP	Torrent
	1	HTTPS	4841	0	2	0
	2	DNS	0	1478	0	0
	3	HTTP	0	0	1050	0
	4	Torrent	0	4	0	62

По данной таблице можно видеть, что в 99.9 процентах случаев предсказание классов происходит верно, а это хороший результат проделанной работы.

Алгоритм «Случайный Лес» как уже говорилось в начале имеет возможность определять значимость признаков в модели, так вот он может предоставить нам информацию о том, какие признаки оказались наиболее значимыми, в отличие от других. Для этого была составлена табл. 5, где показаны признаки, на которые опирался алгоритм и их важность. Так как признаков много, в таблице представлены некоторые из них.

Таблица 5

Количество потоков в выборках

№ п/п	Признак	Важность
1	Пакеты со стороны клиента (кол-во)	0.1334
2	Средний размер части данных со стороны клиента	0.1178
3	Размер первого пакета со стороны клиента	0.1022
4	Кол-во переданных клиентом байт	0.0811
5	Отклонение размера пакета со стороны клиента	0.0783

По табл. 5 можно определить, что на первых строчках указаны признаки больше всего связанные с клиентами, а значит, что серверы приложений не так сильно отличаются друг от друга чем их клиентская часть.

Заключение

Данная статья посвящена разработке и реализации классификации трафика на основе машинного обучения. В отличие от сигнатурных методов, машинное обучение позволяет производить классификацию трафика наиболее точно, количество ложных срабатываний минимально, выявляет аномалии. Также можно отметить, что выбор атрибутов для классификации трафика компьютерной сети – это сложная задача, которая зависит от цели классификации. Уже по малому количеству переданных данных имеем возможность определить протокол прикладного уровня.

Список литературы

1. Лапонина, О. Р. Hassan. Межсетевые экраны / Лапонина О. Р. – М. : НОУ «Интуит», 2016. – 465 с.

2. VDocuments [Электронный ресурс] / M. Chris Riley and Ben Scott : Deep Packet Inspection: The end of the internet as we know it? – Режим доступа : https://vdocuments.site/deep-packet-inspection-very-well-explained.html

3. BuiltIn [Электронный ресурс] / Niklas Donges : A complete guide to the random forest algorithm. – Режим доступа : https://builtin.com/data-science/random-forest-algorithm

Использование системы поддержки принятия решений в животноводстве

Д. В. Куратник

Студент магистрант

И. В. Илларионов

Доцент

Введение

Основная цель животноводов – содержать животных в наиболее подходящих для них условиях. Их ежедневная работа состоит из множества мелких задач. На основе своих знаний и оценки текущей ситуации животновод должен принимать решения касательно множества вопросов.

На данный момент в животноводстве широко используются специальные программы, позволяющие вносить, хранить данные о животных, и на основе этих данных составлять отчеты. Анализировать такой массив данных, чтобы принять какое-либо решение – задача не по силам животноводу, так как требует больших временных затрат. Руководитель тоже не должен тратить время на то, чтобы вникнуть детально в повседневные проблемы производства, но при этом любой руководитель желает сделать процесс производства более эффективным, с меньшими потерями и затратами.

Проблема заключается в том, что животноводы на местах в рамках своей ежедневной работы могут и должны принимать решения, которые в дальнейшем отразятся на работе всего предприятия. Но у животноводов чаще всего нет возможности отследить и проанализировать всю ситуацию по стаду целиком, и, как следствие, прийти к наиболее верному решению. Помочь в этом могут современные технологии, а именно – системы поддержки принятия решений. Однако на данный момент не существует систем поддержки принятия решений непосредственно для животноводов. Данная работа посвящена разработке системы нечеткого логического вывода для поддержки принятия решений животноводами.

1. Создание системы нечеткого логического вывода

В качестве основы для системы поддержки принятия решений в сфере животноводства предпочтение было отдано применению систем с

нечеткой логикой. Нечеткая логика позволяет оперировать нечеткой информацией, которую сложно представить в количественном виде. Система должна помогать принимать решение относительно коэффициента для расчета количества комбикорма для каждого животного.

При разработке системы нечеткого логического вывода были определены входные и выходные переменные, а также термы, относящиеся к каждой из переменных.

Входными параметрами и их термами являются:

– Масса животного: новорожденные, маленькая, средняя, больше среднего, крупная, взрослые, достаточная, очень крупная.

– Возраст животного: новорожденный, самый младший, младший, средний, старше среднего, старший, взрослый.

Выходным параметром и его термами являются:

– Коэффициент для расчета корма: совсем низкий, низкий, ниже среднего, средний, выше среднего, оптимальный, максимальный.

Для представления базы знаний системы нечеткого логического вывода были выбраны продукционные правила. Был сформулирован набор из продукционных правил, определяющих коэффициент для расчета количества корма для животного. Полученный коэффициент умножается на массу животного, и таким образом рассчитывают количество комбикорма для данного животного. Ниже в качестве примера приведены некоторые правила из базы правил.

1. ЕСЛИ возраст «новорожденный» и масса «новорожденные», ТО коэффициент для расчета корма «очень низкий».

2. ЕСЛИ возраст «новорожденный» и масса «маленькая», ТО коэффициент для расчета корма «очень низкий».

3. ЕСЛИ возраст «маленький» и масса «новорожденные», ТО коэффициент для расчета корма «низкий».

При разработке системы нечеткого вывода необходимо определить алгоритм нечеткого вывода, метод аккумуляции и метод дефазификации. Алгоритм Мамдани был взят как алгоритм нечеткого вывода. В качестве метода аккумуляции был выбран метод max-объединения, метода дефазификации – метод центров тяжести (COG) [1].

Важным аспектом является выбор формы функций принадлежности для каждой из переменных. Для большинства термов переменных были выбраны кусочно-линейные (трапецевидные и треугольные) функции принадлежности [2]. Рассмотрим построение графиков функций принадлежности для термов переменной «Масса». Условно можно разделить степень набора массы животного на несколько термов. Для лучшего понимания, выделим термы, которые

назовем «новорожденные» и «взрослые», хоть речь и идет о весе животного. В таблице представлены названия термов в программе, а также значение массы в кг, которое примерно соответствует данному терму.

Таблица

Термы переменной «Масса»

Терм	**Название терма в программе**	**Числовое значение**
Новорожденные	Newborn	35-42
Маленькая	Little	100-130
Средняя	Average	170-190
Больше среднего	Above_avg	240-260
Крупная	Big	300-360
Взрослые	Grown	360-600
Достаточная	Sufficient	650-750
Очень крупная	Extr_big	>750

В данной работе был использован метод относительных частот для построения функций принадлежности. После сведения в одну таблицу результатов опроса экспертов, был сделан вывод, что для непрерывного представления нечеткой переменной в данном случае подходит Z-образная функция принадлежности с максимумом в a = 20 и минимумом в b = 90 (рис. 1). Аналогичным образом были построены функции принадлежности для всех термов всех переменных. Один из полученных в результате задания функций принадлежности графиков представлен на рис. 2.

В качестве инструмента для проектирования системы нечеткого вывода была выбрана программа Visual FML Tool. Это среда разработки для систем, основанных на нечетком выводе. После внесения в систему базы правил и данных о входных и выходных переменных, был запущен аппарат нечеткого вывода.

2. Результаты нечеткого вывода

При запуске системы нечеткого вывода необходимо задать значения входных переменных (возраст и массу животного). Выходным параметром является коэффициент расчета корма. Например, для параметров масса равная 90 кг, возраст равный 80 дней был получен результат 0,215. При умножении данного коэффициента на массу животного, получим количество корма, равное 19,35 кг.

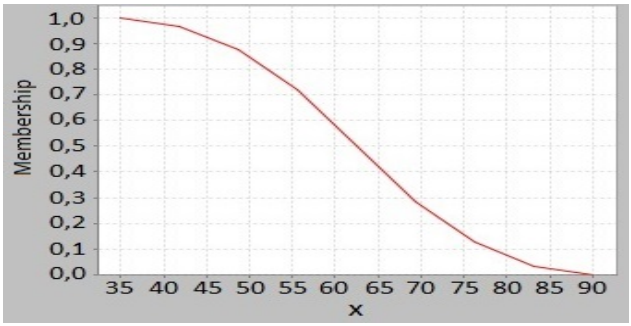


Рис. 1. График функции принадлежности терма «Новорожденные» переменной «Масса»

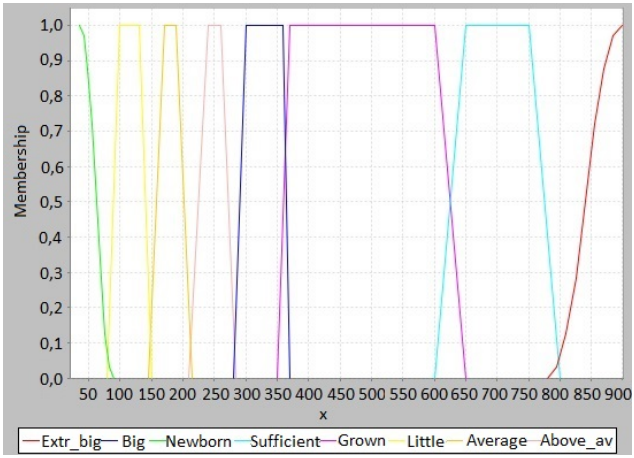


Рис. 2. Графики функций принадлежности всех термов переменной «Масса»

Данный результат отвечает требованиям кормления животных, установленным на ферме. Было выполнено построение поверхности отклика – трехмерного графика, иллюстрирующего поведение нечеткой системы (рис. 3). График поверхности показывает выходную переменную как функцию двух входных переменных. Результаты, полученные по итогам работы системы, близки к расчетам, полученным на основе других методов, что подтверждает их достоверность [3].

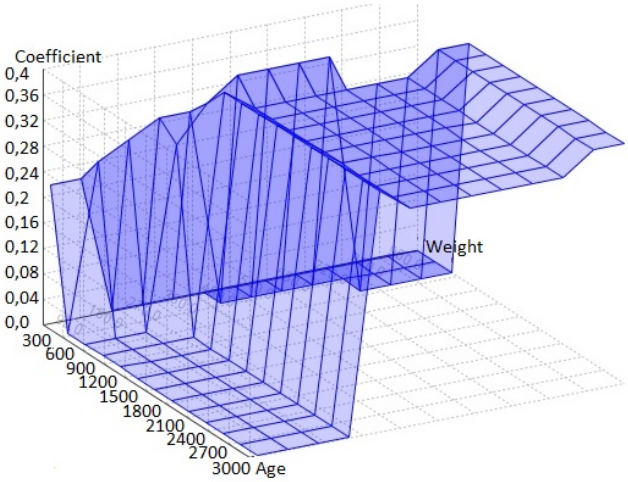


Рис. 3. Поверхность отклика системы

Заключение

Для поддержки принятия решений по количеству комбикорма для крупного рогатого скота была построена система с нечетким выводом. Была сформирована база экспертных продукционных правил и проведена фаззификация входных параметров с использованием заданных форм функций принадлежности. Полученные значения коэффициентов близки к реально используемым на предприятии, что свидетельствует о хорошем качестве построенной системы.

Список литературы

1. Рубанов, В. Г. Интеллектуальные системы автоматического управления. Нечеткое управление в технических системах / В. Г. Рубанов, А. Г. Филатов, И. А. Рыбин. – Белгород : Изд-во БГТУ, 2010. – 170 с.

2. Орловский, С. А. Проблемы принятия решений при нечеткой исходной информации / С. А. Орловский. – М. : Наука, 1981. – 208 с.

3. Чернов, В. Г. Нечеткие контроллеры. Основы теории и построения : учеб. пособие по курсу «Интеллектуальные системы управления» / В. Г. Чернов. – Владимир : Владим. гос. ун-т., 2003. – 148 с.

Система анализа респираторной активности

П. О. Левчук

Студент магистрант

А. Ю. Иванков

Ассистент

А. В. Максимов

Ассистент

Введение

Кашель является самым распространенным симптомом заболеваний органов дыхания и встречается у 20% больных. На данный момент времени диагностика кашля с помощью визуальных аналоговых шкал и специальных опросников отличается выраженной субъективностью.

В связи с этим было принято решение разработать систему анализа респираторной активности. В ходе предыдущих работ было разработано устройство для сбора кашлей, которое активируется при превышении порога звука и движении грудной клетки. Данная работа посвящена исследованию эффективности алгоритмов классификации собранных кашлей с использованием нейронных прямого распространения для использования в системе анализа респираторной активности.

1. Данные

Было собрано 1274 образца одиночных кашлей в формате .wav, частотой дискретизации 41КГц. Среди них 939 сухих кашля, 163 влажных кашля, 172 свистящих кашля.

Для получения признаков для подачи на вход алгоритму мы будем использовать мел-кепстральные коэффициенты. Мел – психофизическая единица высоты звука, основанная на восприятии этого звука органами слуха человека. Это отражает тот факт, что люди гораздо лучше различают небольшие изменения высоты звука на низких частотах, чем на высоких. Так как это психофизическая величина, то существуют разные формулы. Одной из них является формула HTK (см. рис. 1)

$$M(f) = 1127 \cdot \ln(1 + f / 700).$$

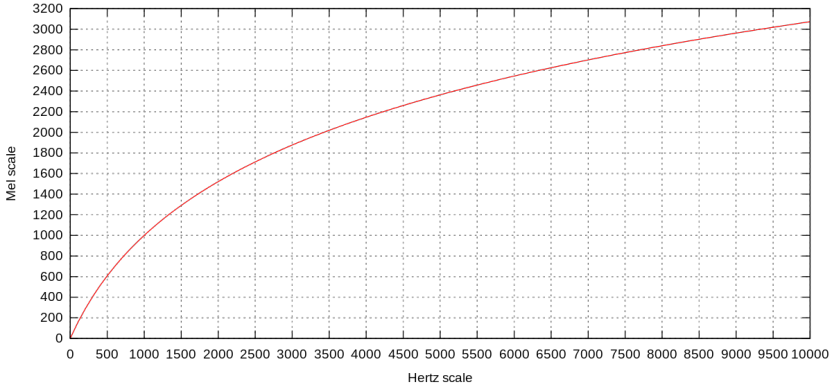


Рис. 1. Шкала перевода из Гц в мел

Для вычисления MFCC необходимо[1]:

1. Разделить сигнал на фреймы. Размер фрема обычно около 20мс для распознавая речи и около 90мс для музыки. Чем меньше размер окна, тем проще различить импульсы, но сложнее различить тона.

2. Применить Short-time преобразование фурье на фреймы, получив спектры.

3. Получить мел-спектрограмму, перемножив спектры на треугольные окна (см. рис. 2).

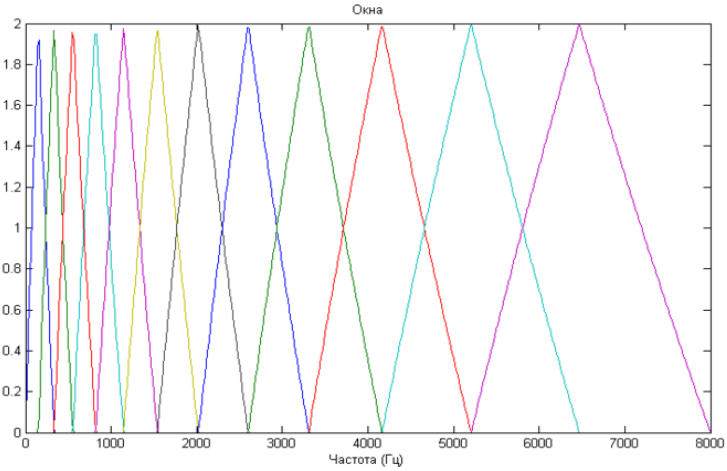


Рис. 2. Треугольные окна в шкале Гц

4. Далее нам необходимо получить мел-кепстр. Кепстр – один из видов обработки сигналов, функция обратного преобразования Фурье от логарифма спектра мощности сигнала. Позволяет описать график спектра относительно небольшим количеством значений.
5. Вычислить логарифм мощностей каждой мел частоты.
6. Применить дискретное косинусное преобразование.
7. Полученные амплитуды спектра это MFCC (см. рис. 3).

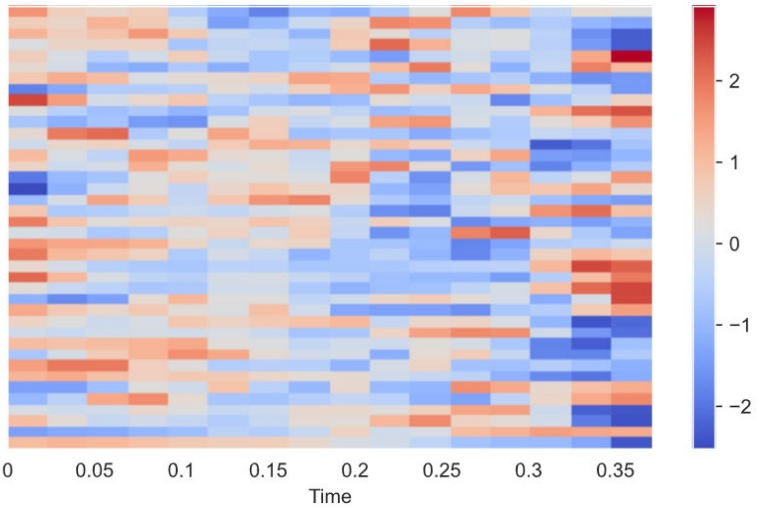


Рис. 3. MFCCs в нормализованном виде

Для нахождения MFCC использовалась библиотека librosa. После получения [N_MFCC; M] массива, где N_MFCC – количество MFCC коэффициентов и M – количество фреймов. Вычисляем min, max, mean, median, std, kurtosis, skewness в строке для каждого коэффицента. Таким образом, получаем 7 * N_MFCC признаков для каждого образца.

2. Обучение

Для изучения эффективности нейронных сетей прямого распространения была разработана модель нейронной сети (см. листинг) с помощью библиотеки pytorch. В качестве функция активации для задачи классификации была выбрана кросс-энтропия с весами 1.0 (Сухой), 3.0 (Мокрый), 3.0 (Свистящий), чтобы учести несблансированность выборки. Были спользованы активация SELU и альфа-исключение [2], оптимизатор AdamW [3].

Архитектура нейронной сети

```
model_args = [
  torch.nn.Linear(feature_count, units),
  torch.nn.SELU(),
  torch.nn.AlphaDropout(dropout),
]
for layer_index in range(layers - 1):
  model_args.extend([
    torch.nn.Linear(units, units),
    torch.nn.SELU(),
    torch.nn.AlphaDropout(dropout)
  ])
model_args.append(torch.nn.Linear(units, n_classes))
return torch.nn.Sequential(*model_args)
```

При обучении использовались несколько случайных разделений выборок на 80% обучающую, 10% валидационную, 10% тестовую. Для увеличения объема обучающей выборки и улучшения устойчивости алгоритма на обучающую выборку был наложен белый гауссовский шум. При подборе гиперпараметров модели и признаков производилось обучение в течение 1000 эпох, из которых выбиралась с модель с наименьшим количеством потерь на валидационной выборке. Вычесления производились на рабочей станции с данными компонентами: AMD Ryzen 9 3950X, Geforce GTX 2070 Super, 2x16Гб 3200МГц ОЗУ, 1Тб SSD Samsung 970 Evo Plus. Подбираемые параметры:

1. Количество коэффициентов MFCC;
2. Размер фрейма;
3. Количество скрытых слоев;
4. Количество нейронов в слое;
5. Дропаут.

Экспериментальным путем мы получили, что для получения подходящих необходим размер фрейма 25мс, количестве вычисленных MFCC коэффицентов от 30, от 5 скрытых слоев с 64 нейронами и дропаутом 0.1. График обучения нейронной сетт с наилучшеми результатами изображена на рис. 4. Матрица запутывания этой сети изображены на рис. 5.

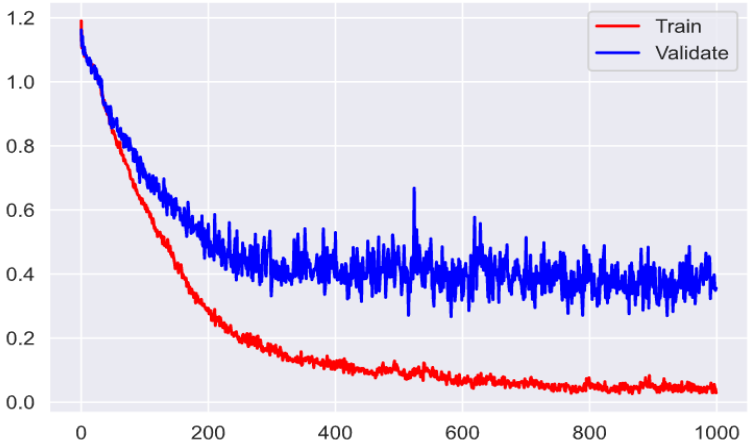


Рис. 4. Потеря/эпоха

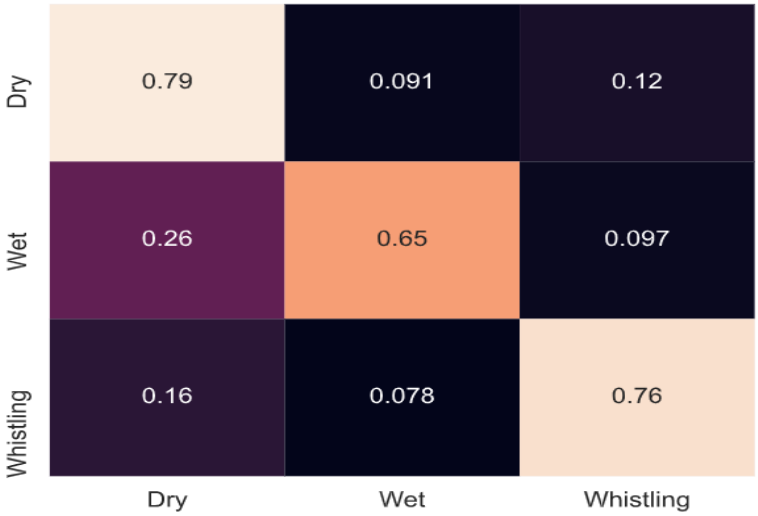


Рис. 5. Матрица запутывания

Заключение

Данная работа посвящена разработке классификатора кашля для использования в системе анализа респираторной активности. В итоге данной разработки мы изучили эффективность нейроных сетей прямого распространения при классификации кашлей.

Дальнейшая работа включает в себя изучение возможности развертки на микроконтроллерах, применение различных способов предобработки информации, создание модели нейронной сети для распознавания кашля и поиск более эффективных алгоритмов [4].

Список литературы

1. Sahidullah, Md. Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition [Электронный ресурс] / Md. Sahidullah, Goutam Saha // Speech Communication. – Режим доступа : 10.1016/j.specom.2011.11.004
2. Klambauer, Günter. “Self-Normalizing Neural Networks. Advances in Neural Information Processing Systems 30 [Электронный ресурс] / Günter Klambauer, Thomas Unterthiner, Andreas Mayr, Sepp Hochreiter. – Режим доступа : https://arxiv.org/abs/1706.02515
3. Loshchilov, Ilya. Decoupled Weight Decay Regularization [Электронный ресурс] / Ilya Loshchilov, Frank Hutter. – Режим доступа : https://arxiv.org/abs/1711.05101
4. CoughRecognition [Электронный ресурс]. – Режим доступа : https://github.com/VSU-CS-MCS/CoughRecognition

Разработка системы составления расписания факультета

М. Б. Леонтьев

Студент магистрант

П. С. Лысачев

Старший преподаватель

Введение

В большинстве случаев расписания на факультетах составляются вручную определенными людьми на основе старых расписаний и опросах преподавателей с помощью Microsoft Excel. Однако, в этом случае зачастую происходят ошибки, вызванные человеческим фактором:

– Учебные аудитории имеют свою собственную вместительность. Человек, составляющий расписание, может неверно рассчитать (или забыть точное количество) студентов в назначенных в аудиторию группах, что, в конечном итоге, приведет к перегрузке аудитории – будет недостаточно рабочих или посадочных мест;

– Некоторые преподаватели совмещают работу в ВУЗе с работой на стороне и не могут присутствовать в определенные часы. Составитель расписания обычно собирает пожелания, в какое время какой преподаватель может вести занятия. Тем не менее, возможны случаи утери подобных списков до или во время составления расписания или просто банального «забывания» о желаемых часах. После таких ошибок преподаватели вынуждены искать в расписании новое свободное и доступное время, уведомлять руководство факультета и т. д.;

– Редкая, но тем не менее, встречающаяся ошибка – из-за большого объема расписания иногда имеют место коллизии, когда несколько преподавателей по ошибке назначены на одно время в одну аудиторию. Это обычно приводит к проблемам, описанным в пункте выше.

1. Исследование и анализ предметной области

На рынке существуют программные решения, направленные на создание расписаний и относительно решающие данные проблемы. Главным ограничивающим фактором повсеместного внедрения подобных систем является сложность настройки (в некоторых случаях необходимо нанимать специалиста из компании-поставщика) и достаточно высокая цена этих решений. Очевидно, что для руководств факультетов наименее затратным выходом будет отказаться от использования подобных решений, нежели постоянно тратить деньги на их содержание.

На основе этих фактов можно сформулировать некоторую задачу. Необходимо разработать систему, с помощью которой можно будет решить или избавиться от выявленных проблем при сохранении легкости использования. Основным важным элементом данной системы должна быть функция экспорта расписания в Excel-файл или ему подобный с целью распечатки бумажной версии расписания.

2. Стек технологий

Технологиями, используемыми при разработке, были выбраны:

– Платформа .NET Framework [1];
– Язык программирования C# [1];
– Технология WPF [2, 3];
– СУБД SQLite.

3. База данных системы

Как было сказано ранее, в качестве СУБД была выбрана SQLite. На следующей диаграмме (рис. 1) показана ее диаграмма.

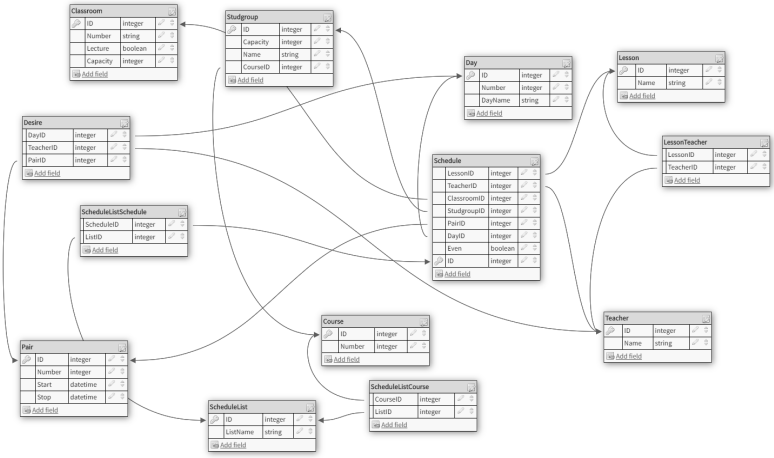


Рис. 1. Диаграмма базы данных системы

Как можно видеть, основной таблицей является Schedule. Другие таблицы Schedule* предназначены для сохранения расписания на листах (которые затем можно экспортировать).

4. Модули системы

Разрабатываемая система состоит из нескольких модулей, изображенных на рис. 2.

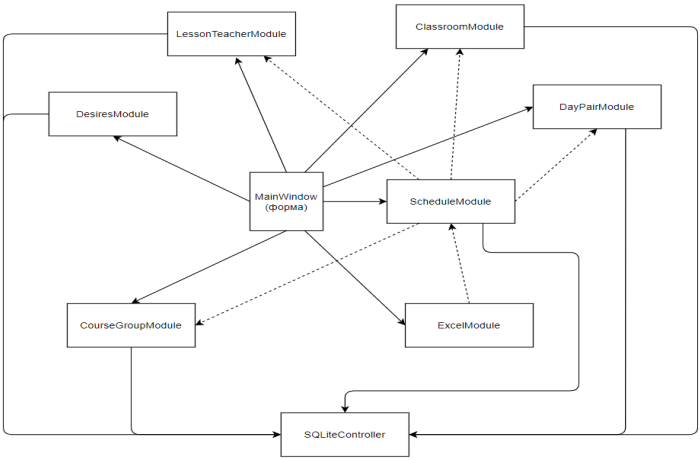


Рис. 2. Схема модулей системы

Как можно видеть, главных модулей в системе три – MainWindow (сама форма с интерфейсом), ScheduleModule – модуль работы с расписанием и SQLiteController, который с помощью ORM-библиотеки отвечает за работу с базой данных.

Остальные модули в определенном смысле «вторичны» и отвечают за определенные действия, ассоциированные со страницами на главной форме. Пунктирными линиями обозначены «слабые взаимодействия» между модулями, которые выражаются в паре-тройке вызовов методов или событий (в основном, для обновления состояния данных).

5. Простой сценарий работы с системой

При описании сценария следует показывать интерфейс разработанной системы. На рис. 3 изображена заполненная страница дней и звонков.

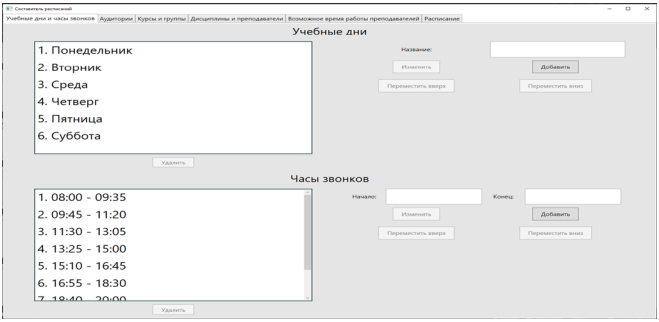


Рис. 3. Добавление дней и звонков

Следом необходимо добавить аудитории (рис. 4).

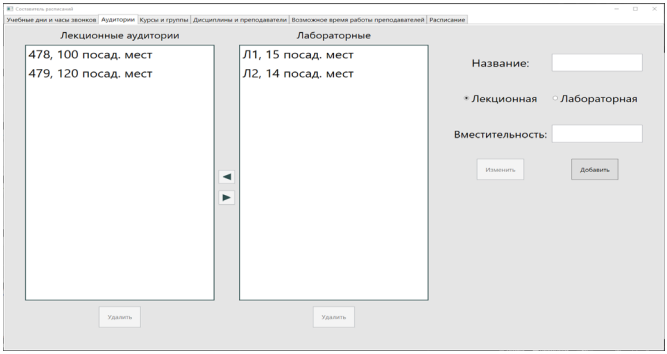


Рис. 4. Добавление аудиторий

Затем необходимо добавить курсы и группы (рис. 5).

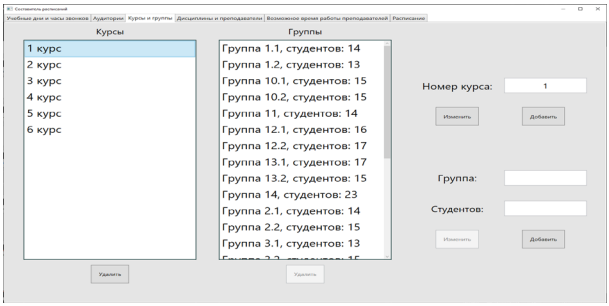


Рис. 5. Добавление курсов и групп

Следующим шагом является добавление дисциплин, преподавателей и установление соответствия (рис. 6).

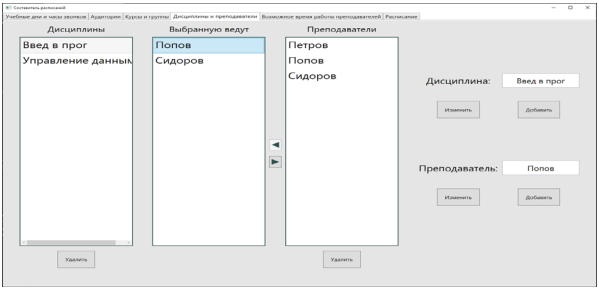


Рис. 6. Добавление дисциплин и преподавателей

Предпоследний шаг — это выбор времени работы (рис. 7).

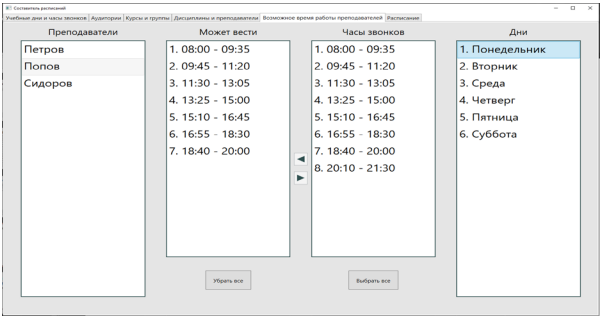


Рис. 7. Выбор времени работы

В конечном итоге необходимо создать лист, выбрать отображаемые на нем курсы и разместить необходимые занятия в соответствующие места (рис. 8). Стоит отметить, что было специально внесено несколько коллизий. В правом верхнем углу страницы с расписанием присутствует область уведомлений.

В аудиторию, вмещающую 15 студентов, было назначено на одно время более 60. В эту же аудиторию было назначено два разных преподавателя. Кроме того, одному из преподавателей было назначено занятие на 20 часов, когда он не имеет возможности его проводить. Всё это в том или ином виде отражено в области уведомлений.

После составления расписания достаточно нажать кнопку «Сохранить» и в итоге будет получен отформатированный файл (рис. 9).

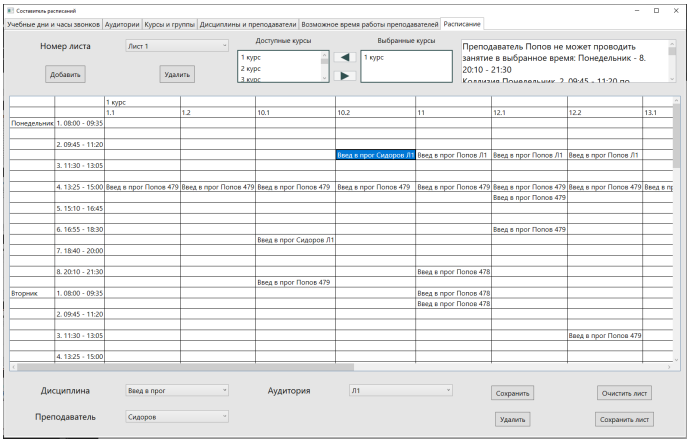


Рис. 8. Заполнение расписания

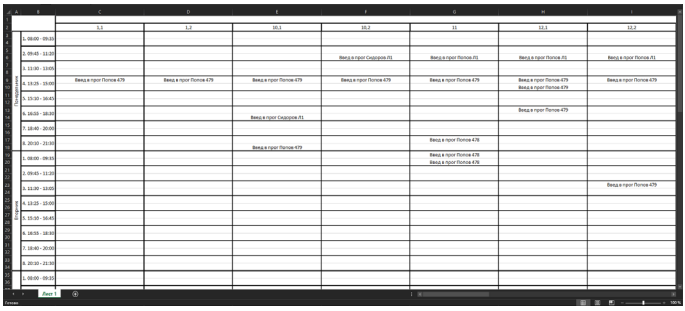


Рис. 9. Экспортированный Excel-файл с расписанием

Заключение

Была разработана система, выполняющая поставленные задачи. Система позволяет создавать расписание и формировать Excel-файл, при этом она постоянно следит за типичными коллизиями, в случае появления оных сигнализирующая в области уведомлений.

В качестве дальнейших направлений усовершенствования и развития данной системы можно выделить следующее:

– Более наглядная цветная отрисовка коллизий;

– Создание веб-версии для возможности просмотра расписания, разделение ролей, личные кабинеты и уведомления, перевод модуля пожеланий в режим привязки к личному кабинету преподавателя для облегчения сбора пожеланий;

– Нейросетевой подход/подход на основе нечеткой логики к автоматизированному составлению расписания;

– Интеграция с системами учета посещаемости;

– Система голосования для студентов и преподавателей;

– Систему связи между студентами и преподавателями;

– Мобильное приложение.

Список литературы

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. Мастер-класс. / Дж. Рихтер ; Пер. с англ. – 2-е изд., исправ. – СПб. : Питер , 2008. – 656 с.

2. Макдональд, М. WPF : Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов. / М. Макдональд ; Пер. с англ. – 4-е изд. – М. : Вильямс , 2013. – 1024 с.

3. XAML в WPF : [Электронный ресурс]. – Режим доступа : https://docs.microsoft.com/ru-ru/dotnet/framework/wpf/advanced/xaml-in-wpf

Исследование проблем индивидуального подхода в обучении на основе программно-целевого подхода

А. Е. Лоскутова

Студент магистрант

В. А. Дурденко

Профессор

Введение

Методом обучения, когда обучающийся не получает знаний в готовом виде, а находит их практическим путем называют деятельностный подход. Именно системно-деятельностный подход является важной частью Федерального Государственного Образовательного Стандарта (ФГОС). Однако только систематизация и интегрирование знаний может справиться с поставленной перед ним задачей [1].

Отсюда вытекает основная цель системно-деятельностного подхода, состоящая в том, чтобы определить тему, которая заинтересует обучающегося, и замотивирует его в большей части изучать материал самостоятельно. Важно дать правильное направление, составить корректный план, в соответствии с которым будет выполняться работа педагога как наставника.

Можно поднять вопрос о том, чем же будет являться этот корректный план, что он будет из себя представлять, кто будет его составлять.

Учитель, педагог, наставник – человек, который контролирует ребенка с самого начала, дает подсказки и советы. Важно не просто заинтересовать ребенка какой-либо темой, но и развить его интерес и навыки для реализации собственного проекта, одного из главных требований обучения в рамках ФГОС. Проектом называется совокупность процессов, состоящих из скоординированных и управляемых задач с начальной и конечными датами, принятыми для достижения цели.

Речь пойдет о работе во внеурочной деятельности, так называем дополнительных занятиях по информатике, которая широко развернута на базе МБОУ «Лицей №1». Проблему и решение будем рассматривать в разрезе элективного курса «Программирование на Python и Java».

Учитель есть некий управленец процессом и результатом, так как по большей части успешность достижения поставленных целей зависит именно от него. Однако учитывая, что у одного педагога на контроле находится достаточно большое количество обучающихся, индивидуальность подхода становится основной проблемой. Решение ее позволяет более детально контролировать процесс и соответственно получать лучшие результаты.

Для получения лучших результатов, достижения желаемых целей используется важный инструмент – программно-целевой подход. Таким образом, подход будет применяться в сфере образования, что само по себе является неким новым решением по его применению. Теперь при построении дерева целей, основной миссией будет являться не получение прибыли, а достижение высоких результатов в IT-конкурсах и соревнованиях.

Любой подход должен заключать в себе некие критерии, по которым происходит расчет эффективности применения. Для подобных расчетов применяется показатель эффективности KPI, индикатор успешности конкретного ребенка «сегодня относительно вчера». То есть в системе происходят постоянный мониторинг текущей ситуации и её оценка.

Таким образом, целью работы стало создание системы для индивидуального подхода в обучении на основе программно-целевого подхода.

1. Исследование области применения и её анализ

Создание системы, которая будет функционировать в рамках дополнительных школьных курсов по программированию, стоит отметить, что речь пойдет именно о работе с детьми. Это важный аргумент, потому что основным требованием будет максимальная компетентность в педагогической и научной сферах одновременно. Это означает, что перед тем как начать работу над каким-либо проектом, необходимо досконально изучить имеющийся уровень знаний ребёнка, стремление к изучению нового материала, выбранную область информационных технологий и возможную сложность проекта. Для этого было решено использовать инструмент – SWOT-анализ, который отражает сильные и слабые стороны процесса обучения, возможности и угрозы получения наилучшего результата (см. таблицу).

Таблица

SWOT-анализ сферы образования

Сильные стороны	**Слабые стороны**
Высокая мотивация учащихся. Первоначальная IT-подготовка (знания языков программирования, полученные на занятиях в урочное и неурочное время). Развитие межпредметных связей.	Отсутствие необходимого количества времени из-за серьезных нагрузок в школе и на дополнительных занятиях. Сильная увлеченность проектом может ухудшить успеваемость по основным предметам в школе.
Возможности	**Угрозы**
Большое количество современных источников. Наличие дополнительных занятий и курсов в школе и высших учебных заведениях.	Наличие серьезных конкурентов на очных олимпиадах и конкурсах по информатике. Малое количество бесплатных олимпиад и конкурсов.

Приступая к составлению индивидуальных учебных планов, учитывалось всё содержимое таблицы. Разработан общий план для конкретного ученика все действия будут адаптированы с учетом специфики его знаний и стремлений к результатам [2].

Для наглядности использована нотация IDEF0, которая используется в методологии структурного анализа SADT. С её помощью легко создается функциональная модель, которая отражает структуру и функции системы, а также наличие потоков информации.

По стандартному методу проектирования первой является контекстная диаграмма. Она проста для понимания, потому что на ней представлен всего лишь один блок, наименование которого будет зависеть от желаемого результата, например, «Победа в конкурсе IT&Kids» или «Призерское место на региональном этапе Всероссийской олимпиады школьников».

На примере цели «Призовое место на региональном этапе Всероссийской олимпиады школьников» был разработан алгоритм подготовки. Как известно, данная олимпиада имеет несколько уровней, сначала школьный, затем муниципальный, региональный и т.д. Для педагога, работающего в профильных классах, конечно, следует брать результаты не ниже регионального уровня.

Поучается следующая диаграмма (рис. 1). Она реализована с помощью приложения ERwin Process Modeler [3]:

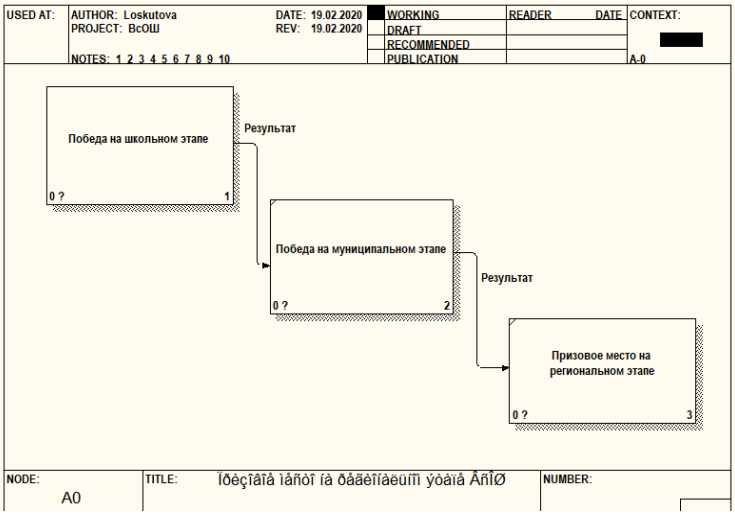


Рис. 1. Общий алгоритм победы

Однако предлагается рассмотреть декомпозицию первой декомпозиции контекстной диаграммы. Это будет уровень диаграммы A1 (рис. 2).

Если подчеркивать именно индивидуальность работы педагога-наставника, то основные различия при одной и той же цели будут на стадии декомпозиции A1 и ниже. Потому что общий «алгоритм победы» действительно будет общим. Под каждого ребенка строится такая диаграмма, которая будет декомпозироваться дальше до простейших минимальных достижений с учетом промежуточных результатов по этим достижениям. Получается индивидуальная рабочая программа, но которая используется не в урочное время, а в неурочное, то есть на дополнительных занятиях. Все достижения ребенка будут собираться в некий архив, который будет демонстрировать развитие его достижений и успехов, что может еще больше замотивировать учащегося [4].

2. Расчет показателя KPI

Достаточно большое внимание уделено способу мониторинга и оценке результатов ученика. Для этого используется мощный инструмент – показатель эффективности KPI. Чаще всего данный показатель используют для просчета эффективности системы на предприятии, однако было решено адаптировать его для проектной деятельности школьного уровня, тем самым подчеркивая его универсальность.

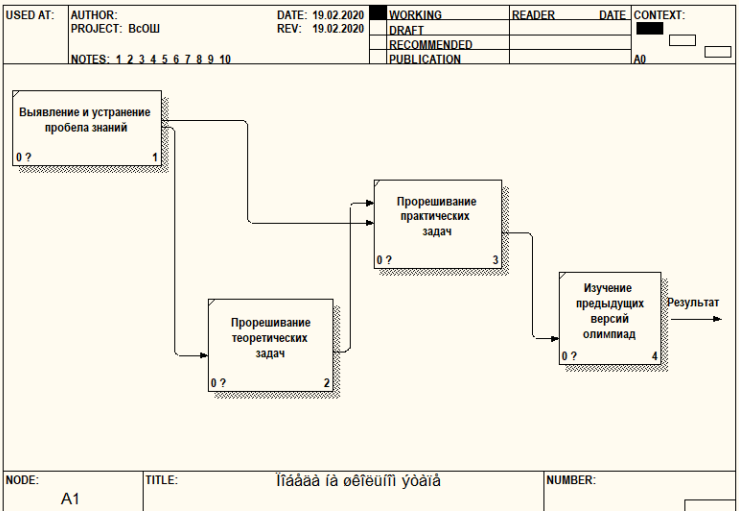


Рис. 2. Общий алгоритм подготовки к олимпиаде

Основная формула расчета индекса *KPI*:

$$KPI = (Факт - База) / (Норма - База) * 100\% .$$

Используя формулу расчета индекса *KPI* можно рассчитать эффективность мероприятий, так как необходимо выбрать несколько ключевых показателей эффективности и просчитать общий результат (рис. 3).

Ключевые показатели	**Веса**	**База**	**Норма**	**Цель**	**Факт**	**Индекс KPI**
Качество оформления документов и проекта	0,3	0	80%	100%	75%	93,75%
Оперативность исправления ошибок	0,3	0	80%	100%	90%	112,50%
Отношения с коллегами по проекту	0,1	0	80%	100%	50%	62,50%
Своевременность промежуточной сдачи результатов	0,3	0	80%	100%	90%	112,50%
	Коэффициент результативности:					101,88%

Рис. 3. Расчет KPI конкретного учащегося

3. Результаты внедрения системы

Наблюдая и занимаясь подготовкой учащихся в течении нескольких лет была выявлена тенденция к цифровизации данного процесса. Этапы Всероссийской олимпиады школьников проходят на муниципальном уровне и выше через площадку Яндекс.Контест. Это отражает необходимость в ведении более гибкой политики в обучении, направленной на работу именно с онлайн-источниками, IT-технологиями и различными автоматизированными системами.

В 2018-2019 учебном году, учащийся 5-ого класса Лоскутовой А. Е. Лопухинский Е. Ю. стал победителем Всероссийской олимпиады школьников по информатике на школьном уровне, однако пройти дальше не представлялось возможным из-за ограничений по возрасту. Только с 9 класса можно переходить на муниципальный уровень и выше. Наблюдая за его успехами в работе было предложено участвовать в различных онлайн-олимпиадах с образовательного портала Учеба.ру, как результат – дипломы победителя в таких олимпиадах, как «Солнечный свет» и «Фоксфорд». Продолжая взаимодействовать с родителями учащегося, было решено углубить знания в языке программирования Scratch параллельно выстраивая первые варианты рабочей системы используя технологию IDEF0. Была выдвинута идея о создании образовательного проекта по правилам дорожного движения. Данный проект представлял собой игру. С данным проектом Лопухинский Е. Ю. выступил в двух региональных конференциях:

1. XVII региональная научно-практическая конференция «Путь в науку». Результат – диплом II степени.
2. XXXIV конференция НОУ ВГУ. Результат – диплом II степени.

Еще одним важным достижением 2018-2019 учебного года стала работа ученицы 11-ого класса Кудиновой М. Ю., с которой работа велась непосредственно на занятиях по внеурочной деятельности. Критериями оценки успешности работы стали не только дипломы и сертификаты, полученные на конференциях, но также и расчет показателя KPI, который постепенно увеличивался. Обучающаяся под чутким руководством Лоскутовой А. Е. выполнила ряд проектов, связанных с написанием различных игровых программ на разных языках программирования. Итогом этого года стали:

1. Диплом I степени, полученный на XVII региональной научно-практической конференции «Путь в науку».
2. Диплом II степени, полученный на XXXIV конференция НОУ ВГУ.

3. Диплом I степени, полученный на региональном конкурсе компьютерных проектов IT&Kids, который проводился на базе ФКН ВГУ.

Несмотря на то, что 2019-2020 учебный год еще не закончен, но следуя системе с индивидуальным подходом для одаренных детей уже получены следующие результаты в олимпиадах:

1. ВсОШ, Саликова М. Т., учащаяся 8 класса, стала победителем.
2. ВсОШ, Слинчук Д. А, учащийся 9 класса, стал призером.
3. ВсОШ, Светличный Л. А, учащийся 9 класса, стал призером.

Особое значение уделено учащемуся 9 класса Белянин Г. О., который стал победителем школьного этапа ВсОШ, затем стал призером на муниципальном и региональном этапах. Подготовка занимала по одному академическому часу в неделю в рамках внеурочной деятельности. Для успешности его деятельности применена система, где все действия ребёнка и педагога-наставника описаны с помощью нотации IDEF0 и EPC.

В данный период времени активно идет разработка проектов для участия в XVIII научно-практической конференции «Путь в науку» и конференции НОУ ВГУ, а также участие в олимпиадах по теоретической информатике, организованной на ФКН ВГУ и «Смарт Старт».

На данный момент времени для апробации первых проектов было решено участвовать в Всероссийском конкурсе юных инженеров-исследователей среди одаренных школьников для 7-11 классов, которая проводится под патронатом Самарского национального исследовательского университета имени академика С.П. Королева. Финалистом данного конкурса в направлении «Информационные технологии» стала ученица 8-ого класса Саликова М. Т. Этот результат подтверждает успешность и целесообразность разработанной системы.

Заключение

В результате работы системы, внедренной в работу учителя общеобразовательной школы с углубленным изучением информатики можно заметить следующую особенность. При сравнении полученных результатов Всероссийской олимпиады школьников за 2018-2019 и 2019-2020 учебный год заметна тенденция к улучшению результатов. Это связано с тем, что учащийся сам представляет собой проект для педагога. Как к любому проекту, к каждому ребенку требуется индивидуальный подход. Именно это реализовано в системе. Происходит индивидуальная разработка не только материала для изучения, но и подбор мероприятий, в которых ребенок может себя проявить. Учитель есть менеджер, ответственный за результат. Успехи учащегося являются напрямую успехами его учителя, при условии их

постоянного сотрудничества и мониторинга промежуточных результатов.

Список литературы

1. Лоскутова, А. Е. Системно-деятельностный подход в обучении как инструмент при решении задач повышенной сложности по информатике на языке программирования Java / А.Е. Лоскутова // Инновационная школа: проблемы и пути их решения: сборник научно-методических статей / под ред. О. Е. Струковой, Т. Е. Лапшиной, Э. П. Жарких. – Воронеж : ООО «РИТМ», 2019. – С.54-58.

2. Черемных, С. В. Моделирование и анализ систем. IDEF-технологии: практикум / С. В. Черемных, И. О. Семенов. – М. : Финансы и статистика, 2006. – 192 с.

3. Erwin [Электронный ресурс]. – Режим доступа : https://erwin.com/products/erwin-evolve/

4. Методология функционального моделирования IDEF0. – М. : ИПК Издательство Стандартов, 2000. – 75 с.

Реконструкция цифровых изображений методом частичной свертки

Н. А. Лукин

Студент магистрант

М. А. Дрюченко

Доцент

Введение

В наши дни существует множество различных способов цифровой обработки изображений, например, размытие, повышение резкости, нахождение краев, тиснение и прочие. Наличие большого количества методов обработки в большей степени обусловлено наличием разного вида дефектов, возникающих при работе с цифровыми изображениями. Это несовершенство регистрирующих приборов, вследствие чего возникают шумы на изображении, представленные на рис. 1а, а также сильное сжатие с потерей качества, используемое для хранения и передачи изображений, результатом чего является нечеткость, размытие, потеря фрагментов (рис. 1б, 1в). Однако, несмотря на наличие возникающих трудностей, сфера обработки цифровых изображений

остается одной из самых востребованных и развивающихся. Причинами этого является широкая область применения данных технологий, в числе которых, например, распознавание текстов, машинное зрение, обработка спутниковых снимков, идентификация личности, автоматическое управление автомобилями и другие. Всё это становится возможным в связи с повышением производственных мощностей и расширяющимся спектром функций систем обработки, хранения и передачи информации, а также появлением новых технологий, методов обработки и анализа информации, что в свою очередь поддерживает уровень развития методов обработки цифровых изображений.



а б в

Рис. 1. Причины и примеры шума изображения:
а – несовершенство регистрирующих приборов, б – хранение и передача изображений с потерей данных, в – нечеткость изображения, явно выраженные пиксели

Итак, способы обработки цифровых изображений, которые были перечислены, а так же множество других основаны на свертке. Что же такое свертка? Свертка (англ. convolution) — это операция, показывающая «схожесть» одной функции с отражённой и сдвинутой копией другой. Понятие свёртки обобщается для функций, определённых на группах, а также мер. В случае обработки изображений свертка фактически сводится к операции вычисления нового значения выбранного пикселя, учитывающая значения окружающих его пикселей. Одним из методов связанных с восстановлением изображений, использующих в своей основе свертку, является метод PatchMatch, имеющий определенные недостатки, вследствие чего появилась возможность использования метода частичной свертки, который имеет ряд преимуществ над PatchMatch.

Целью данной работы и является реконструкции цифровых изображений методом частичной свертки [1].

1. Заполнение пробелов в изображении

Задача заполнения пробелов в изображении (англ. – «Image inpainting») — состоит в создании модели, которая надежно работает на нерегулярных пробелах и создает семантически значимые прогнозы, которые сочетаются с остальной частью изображения без необходимости дополнительных операций пост-обработки или смешивания. Модель используется со многими приложениями, например, при редактировании изображений для удаления нежелательных деталей в изображении, заполняя их приемлемым контентом.

Для данной задачи используется множество подходов, но ни один из них не использует метод глубокого обучения, и у них есть недостатки. Один из методов называется patchmatch, который итеративно ищет подходящие исправления для заполнения пробелов. Хотя этот подход в результаты дает плавный переход, он ограничен доступной статистикой изображений и не использует понятие визуальной семантики. Другое ограничение подобных подходов — фокусировка на прямоугольных пробелах, которые часто считаются центральными в изображении [2]. Ограничения приводят к перенасыщению прямоугольных пробелов и, в конечном счете, ограничивают применение этих моделей.

2. Метод частичной свертки

Одним из вариантов снятия ограничения предыдущего подхода, является использование частичной свертки для решения проблемы зарисовки изображения. Частичный сверточный слой содержащий операцию маскировки и повторную нормализацию свертки, следуют за обновлением маски. Основным расширением является этап автоматического обновления маски, который удаляет маскировку, где частичная свертка работала по незамаскированному значению.

Новшества данного метода:

– Использование частичных сверток с автоматическим шагом обновления маски для достижения современных результатов в восстановлении изображений (image-inpainting).

– При помощи замены сверточных слоев частичными свертками и обновления масок удалось достичь самых высоких результатов в реконструкции изображений.

– Эффективность обучения image-inpainting моделей на пробелах неправильной формы.

Модель использует операцию частичной свертки и функция обновления маски, которые, работая совместно, образуют частичный сверточный слой [3].

Пусть W — вес фильтра свертки; b — соответствующее смещение; X — значения пикселей для текущего окна свертки; M — соответствующая бинарная маска. Частичная свертка в каждом месте выражается следующим образом:

$$x' = \begin{cases} \mathrm{W}^{\mathrm{T}}\left(\mathrm{X} \otimes \mathrm{M}\right)\dfrac{1}{\mathrm{sum(M)}} + \mathrm{b}, \text{ if } \mathrm{sum(M)} > 0 \\ 0, \qquad \text{otherwise} \end{cases} \tag{1}$$

где $\otimes$ обозначает умножение по элементам.

Как видно, выходные значения зависят только от немаскируемых входных значений. С помощью масштабного коэффициента 1/sum(**M**) подбирается подходящее масштабирование, чтобы скорректировать изменяющиеся действительные (незамаскированные) входные данные. После каждой операции частичной свертки маска обновляется. Правило unmasking простое: если с помощью свертки получилось вывести результат, по крайней мере, для одного допустимого входного значения, тогда маска для этого местоположения удаляется. Это выражается следующим образом:

$$m' = \begin{cases} 1, & \text{if } \mathrm{sum}\left(M\right) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

и легко реализуется в любой системе глубокого обучения как часть forward pass.

3. Сравнение методов обработки изображений

Большинство методов обработки и восстановления цифровых изображений направлено в основном на работу с изображениями, у которых отсутствующие элементы прямоугольной формы, а также расположены ближе к центру изображения. Кроме того, большинство из них требуют ресурсозатратной постобработки. Метод частичной свертки представляет собой улучшенный способ восстановления цифровых изображений, который надежно работает с нерегулярными формами пробелов и дает семантически значимые прогнозы, которые плавно объединяются с остальной частью изображения без необходимости какой-либо дополнительной операции постобработки или смешивания. Согласно проведенным исследованиям частичная свертка превосходит другие методы обработки изображений по многим показателям эффективности алгоритмов, среди которых L1-ошибка, пиковое

отношение сигнала к шуму, индекс структурного сходства, а также начальная оценка. Метод PConv превосходит другие методы в этих измерениях на нерегулярных масках [4].

И действительно, результаты тестирования данного алгоритма на наборах данных Celeba-HQ и ImageNet (рис. 2, 3) демонстрируют, что метод частичной свертки не имеет тех же недостатков, что и предыдущие подходы, для которых в большинстве случае необходима корректировка выходного изображения, чтобы гарантировать, что заполненные пробелы не будут выглядеть неестественно. Также удалось избежать дефектов, таких как зернистость и размытые края. По сути, при использовании данного метода нейронная сеть создает маски и предсказания, основанные на частичной свертке, создавая, таким образом, промежуточный слой, которым можно управлять, пока изображение не пройдет проверку подлинности сетью дискриминатором.

а б в г

Рис. 2. Сравнение метода частичной свертки с методом Contextual Attention: а – исходное изображение, б – метод Contextual Attention, в – метод частичной свертки, г – исходное изображение

а б в г д е

Рис. 3. Сравнение метода частичной свертки другими методами обработки изображений: а – исходное изображение, б – Patch-match, в – GL Consistent Completion, г – Contextual Attention, д – частичная свертка, е – исходное изображение

Использование частичного сверточного слоя с автоматическим механизмом обновления маски и достижения современных результатов в зарисовывании изображений. Модель может надежно обрабатывать дефекты любой формы, размера или расстояния от границ изображения. Кроме того, производительность не ухудшается, если размеры пробелов увеличиваются, как видно на рис. 4.

Рис. 4. Результат работы метода частичной свертки при увеличении размера пробелов

Заключение

Данная статья посвящена описанию метода восстановления изображений с помощью частичной свертки. Изложены основные принципы работы алгоритмов методов обработки с помощью частичной свертки и patchmatch. Была описана функция потерь для алгоритмов глубокого обучения, а также выведена общая функция потерь. Продемонстрированы основные преимущества и недостатки обоих методов, а также проведен сравнительный анализ эффективности других методов обработки цифровых изображений, в результате чего был сделан выбор в пользу PConv – метода частичной свертки. Использование данного решения позволяет реконструировать изображения с нерегулярными формами пробелов.

Список литературы

1. Селянкин, В. Компьютерное зрение. Анализ и обработка изображений / В. Селянкин. – Санкт-Петербург : Лань, 2019. – 152 с.

2. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.

3. Бхуян, М. К. Компьютерное зрение и обработка изображений / М. К. Бхуян. – Boca Raton : CRC Press, 2019. – 442 с.
4. Реконструкция фотографий методом частичной свертки от Nvidia [Электронный ресурс] : сайт. – Режим доступа : https://medium.com/@neurohiveru/реконструкция-фотографий-методом-частичной-свертки-от-nvidia-70ee1fa010f5

Рекомендательная система онлайн заказов в ресторанах

А. И. Маслов

Студент магистрант

А. Ю. Иванков

Ассистент

Введение

В настоящее время существует большое количество различных ресторанов, кафе и закусочных с долгим временем ожидания заказа. При покупке еды в час пик, например, в обед время ожидания может сильно затянуться. Из-за этого люди, которые ограничены временем обеда не всегда могут попасть в желаемое заведение, а значит ресторан теряет потенциальных посетителей. Данная работа посвящена разработке и программной реализации сервиса для онлайн оформления заказа и рекомендации пользователя заведений, которые его могут заинтересовать на основе его предыдущих действий.

1. Система онлайн заказа

Для решения проблемы разработан веб-сервис по агрегации различных ресторанов и кофе. Сервис избавит клиентов заведений от длительного ожидания, обеспечит больше заказов для ресторанов, позволив заранее приготавливать еду для посетителей.

Функции для посетителей:

– Просмотр ближайших заведений с возможностью фильтрации по названию.

– Просмотр меню с указанием цены, различных опций и наличия в выбранном ресторане.

– Возможность сделать предварительный заказ на определенное время или как можно быстрее с оплатой при получении или онлайн.

– Добавление ресторанов в избранное для совершения быстрой покупки в будущем.
– Функции для администраторов ресторана.
– Просмотр списка текущих заказов.
– Возможность принять или отклонить заказ
– Управление статусом заказа.
– Управление наличием позиций в меню.

2. Технологии

Система состоит из двух частей: серверная часть, для обработки всех запросов и клиентской части для создания интерфейса.

Для реализации серверной части используется платформа .net core и язык программирования C#. Net core [1] – это модульная платформа для разработки программного обеспечения с открытым исходным кодом. Совместима с такими операционными системами как Windows, Linux и macOS. Была выпущена компанией Microsoft. **C#** (произносится си шарп) – объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft. C# относится к семье языков с C-подобным синтаксисом. Язык имеет статическую типизацию [2], поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Для реализации клиентской части используются html, css, js и фронт-энд фреймворк vue. HTML – стандартизированный язык разметки документов во Всемирной паутине. CSS – каскадные таблицы стилей) – формальный язык описания внешнего вида документа, написанного с использованием языка разметки. JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Vue.js— JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

3. Аутентификация в приложении

Аутентифика́ция – процедура проверки подлинности, например:

– проверка подлинности пользователя путём сравнения введённого им пароля (для указанного логина) с паролем, сохранённым

в базе данных пользовательских логинов;

– подтверждение подлинности электронного письма путём проверки цифровой подписи письма по открытому ключу отправителя;

– проверка контрольной суммы файла на соответствие сумме, заявленной автором этого файла.

Все взаимодействие с сервисом происходит через запросы к серверу. Для идентификации пользователей используются json web token. JSON Web Token (JWT) – это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

На рис. 1 приводится пример токена следующем формате header.payload.signature. Для его создания необходимо определить заголовок (header) с общей информацией по токену, полезные данные (payload), такие как id пользователя, его роль и т.д. и подписи (signature).

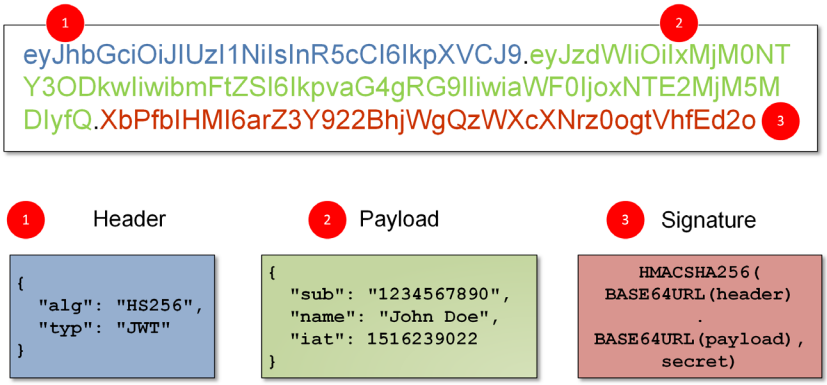


Рис. 1. Json web token

Процесс создания токена:

1. Создаем HEADER. Поле `type` показывает что это JSON Web Token. Поле `alg` определяет алгоритм хеширования `HS256` (для его вычисления нужен лишь один секретный ключ). Он будет использоваться при создании подписи.

2. Создаем PAYLOAD. Payload – это полезные данные, которые хранятся внутри JWT.

3. Создаем SIGNATURE. Алгоритм base64url кодирует хедер и payload, созданные на 1 и 2 шаге. Алгоритм соединяет закодированные

строки через точку. Затем полученная строка хешируется алгоритмом, заданным в хедере на основе нашего секретного ключа.

Очень важно понимать, что использование JWT НЕ скрывает и не маскирует данные автоматически. Причина, почему JWT используются – это проверка, что отправленные данные были действительно отправлены авторизованным источником. Как было продемонстрировано выше, данные внутри JWT закодированы и подписаны, это не одно и тоже, что зашифрованы. Поэтому в токенах нельзя передавать конфиденциальную информацию.

Поскольку приложение знает секретный ключ, когда пользователь делает API-запрос с приложенным к нему токеном, приложение может выполнить тот же алгоритм подписывания к JWT, что в шаге 3 процесса создания токена. Приложение может потом проверить эту подпись, сравнивая ее со своей собственной, вычисленной хешированием. Если подписи совпадают, значит JWT валидный, т.е. пришел от проверенного источника. Если подписи не совпадают, значит что-то пошло не так – возможно, это является признаком потенциальной атаки. Таким образом, проверяя JWT, приложение добавляет доверительный слой между собой и пользователем.

При использовании JSON-токенов в клиент-серверных приложениях реализована следующая схема:

1. Клиент проходит аутентификацию в приложении (к примеру, с использованием логина и пароля).
2. В случае успешной аутентификации сервер отправляет клиенту access(JWT) и refresh токены.
3. При дальнейшем обращении к серверу клиент использует access-токен. Сервер проверяет токен на валидность и предоставляет клиенту доступ к ресурсам.
4. В случае, если access-токен становится не валидным, клиент отправляет refresh токен, в ответ на который сервер предоставляет два обновленных токена.
5. В случае, если refresh-токен становится не валидным, клиент опять должен пройти процесс аутентификации.

В работе с запросами используется этот подход. При обращении клиента на сервер, из токена извлекается идентификатор пользователя и далее все действия происходят от его имени. Для пользователей, которые управляют рестораном, в токене дополнительно хранится роль «Manager» и идентификатор конкретного ресторана. Поэтому при запросе, например, к /api/manager/getCurrentOrders из токена будет извлечена роль и идентификатор управляемого ресторана, и вернутся данные для конкретного заведения.

4. Рекомендательная система

Коллаборативная фильтрация (на соседстве) – это метод построения рекомендаций в рекомендательных системах, которые используют предпочтения группы пользователей для того, чтобы спрогнозировать предпочтения для другого пользователя [3]. Работает коллаборативная фильтрация по следующему принципу: пользователи выражают своё мнение о неких продуктах или услугах на сайте – дают им свою оценку, оставляют отзывы. В итоге можно сформировать группу пользователей с одинаковыми интересами – им нравятся одни товары или услуги, но совершенно не нравятся другие. Проще говоря, у этой группы пользователей будут одни и те же вкусы.

Входные данные: матрица оценок, выставленных пользователями продуктам (рис. 2).

		Items					
		1	*2*	...	*i*	...	*m*
Users	*1*	5	3		1	2	
	2		2				4
	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		
	a	3	5		?	1	

Рис. 2. Матрица оценок

На первом этапе для каждого пользователя вычисляется мера его близости к пользователю a (Коэффициент корреляции Пирсона).

$$W_{a,u} = \frac{\sum_{i \in I} (r_{a,i} - \overline{r_a})(r_{a,i} - \overline{r_i})}{\sqrt{\sum_{i \in I} (r_{a,i} - \overline{r_a})^2 (r_{u,i} - \overline{r_u})^2}},$$

где $W_{a,u}$ – мера близости пользователями a и u, I – множество объектов, оцененных как пользователем a, так и пользователем u, $r_{a,i}$, $r_{u,i}$ – оценка объекта i пользователем a и u, $\overline{r_a}, \overline{r_u}$ – средняя оценка пользователем a и u.

Выбираются пользователи, наиболее близкие к пользователю a (топ k пользователей, порог меры близости).

Предсказывается рейтинг объекта на основе рейтингов выбранных пользователей по формуле в виде

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \subset k} (r_{u,i} - \bar{r}_u) * W_{a,u}}{\sum_{u \subset k} |W_{a,u}|},$$

где $p_{a,i}$ – предсказание оценки, $W_{a,u}$ – мера близости между пользователем a и u, $\bar{r}_a$, $\bar{r}_u$ – средняя оценка пользователем a и u.

Заключение

Данная статья посвящена разработке и реализации рекомендательного сервиса агрегации ресторанов и оформления онлайн заказов. Описаны алгоритмы рекомендательных систем. В итоге данной разработки получился полностью рабочий сервис, реализованный в виде тестового веб-приложения. Использование данного приложения позволяет просматривать список ресторанов, изучать их меню, оформлять заказ онлайн, получать рекомендации на основе прошлых заказов.

Список литературы

1. Рихтер, Д. CLR via C# / Д. Рихтер. – М. : Питер, 2017. – 898 с.
2. Троелсен, Э. Платформы .NET и .NET Core / Э. Троелсен. – М. : Вильямс, 2018. – 1328 с.
3. Гомзин, А. Г. Системы рекомендаций: обзор современных подходов / А. Г. Гомзин, А. В. Коршунов // Труды Института системного программирования РАН. – 2015. – Т. 27. – С. 401-419.

Развитие бизнеса путем повышения клиентского спроса. Метод QFD

В. Г. Матющенко

Студент магистрант

И. В. Абрамов

Доцент

Введение

В современных условиях в России происходит усиление конкуренции во всех сферах экономики. В первую очередь это особенно

видно на предприятиях торгового бизнеса, это ставит перед ними задачу глубокого исследования запросов потребителей и воплощение их желаний в продаваемых товарах и предоставляемых услугах. Только выходя на рынок с товарами и услугами, отвечающими всем запросам потребителя можно обеспечить повышение клиентского спроса и тем самым обеспечить развитие своему бизнесу. Руководители находятся в постоянных поисках новых инструментов управления бизнесом и рычагов повышения конкурентоспособности с целью привлечения покупателя.

Для адаптации системы продаж к покупательскому спросу проводится аналитическая работа по изучению запросов покупателей и воплощению их желаний в продаваемой продукции. Решение этой маркетинговой задачи позволяет продавцу выйти на соответствующий рынок с предложением, которое уже ожидает потребитель и тем самым обойти своих конкурентов в «борьбе» за покупателя. В этом поможет современная методология QFD – Quality Function Deployment. Дословный перевод англоязычного названия метода означает, что речь идет о «развертывание функции качества» (однако в России более известен вариант перевода, сформулированный Ю. П. Адлером [1], – «структурирование функции качества». Несмотря на английское название, методология имеет японское происхождение. Методология QFD (матричное планирование продуктов, матрица решений) позволяет внимательно прислушаться к голосу Клиента, а затем быстро и правильно среагировать на реальные потребности и ожидания потребителей [2].

Если в промышленном производстве метод QFD позволяет спланировать к выпуску продукцию, разработанную с учетом потребностей и ожиданий будущих потребителей, то в сфере торгового бизнеса он поможет правильно спланировать расходную часть бюджета и организовать работу торгового предприятия с учетом приоритетных требований покупателя к товарам и услуга. Данная работа посвящена применению метода QFD в сфере торговли с целью развития бизнеса путем повышения клиентского спроса и расширения рынков сбыта.

1. Применение методологии QFD для идентификации требований потребителя

В современном виде QFD – это математический способ выяснить реальные пожелания потребителя и уровень оценки продукции, уже представленной на рынке [3]. С помощью этого метода продавец может точно понять, совпадает ли его представление о качестве продаваемых товаров с ожиданиями покупателей.

Полезность метода QFD заключается в определении наиболее важных проблем и связанных с ним приоритетов. Он позволят перевести большой объем информации в сжатый и удобный для проведения эффективного и четкого анализа вид, и обеспечить точное доведение потребностей потребителя до производителя или поставщика товаров и услуг.

Для исследования запросов потребителей проводится большая аналитическая работа, результаты которой находят свое отражение в итоговой матричной диаграмме – «Доме качества» (таблица).

«Левая часть» Дома качества – информация о выявленных и проранжированных требованиях потребителей

«Крыша» Дома качества отображает взаимосвязи между требованиями потребителей и статьями расходов для выполнения этих требований.

«Центральная часть» Дома качества представляет собой матрицу $\left\| R_{ij} \right\|$. Строки матрицы (i) заполняются требованиями потребителей. Столбцы матрицы $\left(j \right)$ заполняем различными статьями расходов. Ячейки матрицы отражают коэффициент попарной корреляции R_{ij} между требованием (с номером i) и статьей расходов (с номером j).

Коэффициент попарной корреляции R – это число, абсолютное значение которого изменяется от «0» до «1» и показывает силу корреляционной взаимосвязи между требованиями и расходами на их удовлетворение. Ноль будет показывать, что никакой зависимости нет и нецелесообразно увеличивать расходы, так как на степень удовлетворения требования покупателя это не повлияет.

Для расчета абсолютного значения приоритета статей расходов H воспользуемся следующей формулой:

$$H_j = \sum_{i=1}^{i=N} P_i R_{ij} \qquad (1)$$

где P_i – приоритет $i-го$ требования покупателей; R_{ij} – значение коэффициента парной корреляции между j-й статьей расходов и i-м требованием.

«Правая часть» Дома качества содержит результаты потребительского бенчмаркинга и рассчитанных значений показателей:

K_i – показатель, отображающий приоритет для организации в работе с i-м конкретным требованиям к товарам и услугам на данный момент времени на изучаемом рынке.

$K_i^{\text{отн}}$ – показатель, выраженный в процентах, отображает долю в совокупной важности для организации работы над увеличением выполнения всего списка требований покупателей, которая относится к конкретному i-му требованию.

V_i – комплексный нормированный показатель, выраженный в процентах, представляет собой приоритет i-го требования, но уже не для потребителя, а для предприятия с учетом объективной ситуации на рынке на фоне конкуренции.

Для количественной оценки важности для предприятия вкладывать средства в увеличение степени выполнения конкретного требования потребителей, исходя из построенных профилей конкурирующих организаций воспользуемся следующим аналитическим выражением:

$$K_i = (1,2B_{\max} - B_i^0)(QB_{\max} + \sum_{n=1}^{n=Q}(B_i^n - B_i^0)), \qquad (2)$$

где $B_{\max}$ – максимальный балл по применяемой оценочной шкале; B_i^0 – бальная оценка выполнения i-го требования потребителей; Q – количество конкурирующих организаций; B_i^n – бальная оценка выполнения i-го требования потребителей n-ым конкурентом.

Рассчитанное по этой формуле значение K_i дает реальную информацию для фирмы о степени выполнения i-го требования потребителей в сопоставлении с результатами расчета профилей организаций, занесенных в Дом качества (таблица).

$$K_i^{\text{отн}} = \frac{K_i}{\sum_{i=1}^{i=N} K_i} = \frac{(1,2B_{\max} - B_i^0)(QB_{\max} + \sum_{n=1}^{n=Q}(B_i^n - B_i^0))}{\sum_{i=1}^{i=N}(1,2B_{\max} - B_i^0)(QB_{\max} + \sum_{n=1}^{n=Q}(B_i^n - B_i^0))}. \qquad (3)$$

Стоит заметить, что критерий $K_i^{\text{отн}}$ имеет две характерные особенности. Для первой из них характерна безотносительность к приоритету потребительского требования. Он отражает положение организации на изучаемом рынке по отношению к конкурентам.

Вторая особенность – это динамизм показателя $K_i^{\text{отн}}$. Конкурирующие предприятия находятся в постоянном развитии и поиске путей улучшения бизнеса, поэтому мониторинг должен вестись с определенной периодичностью с составлением «профилей» конкурентов

Таблица

Дом качества

Ранг требования	Требования потребителей	Приоритет требования	Расходы на расширение ассортимента по наименованию	Расходы на расширение ассортимента по производителям	Расходы на рекламу и продвижение	Расходы на интернет-магазин	Расход на технические устройства по оплате и их содержание	Расходы на упаковку	Расходы на заработную плату	Расходы на обучение персонала	Расходы на доставку	Расходы на обработку	Арендная плата на дополнительную торговую площадь	Расходы на комунальные услуги	Расходы на вип клиентов	$K_i^{отн}$	V_i
1	Широкий ассортимент по наименованию	20,19	+0,9	+0,3												7,88%	23,72%
2	Ассортимент по производителям	15,45	+0,9	+0,9												7,16%	16,50%
3	Скидки	11,52	+0,1	+0,1	-0,1	+0,3	-0,1	-0,1	-0,3	-0,1	-0,1		-0,1		+0,1	4,30%	7,38%
4	Интернет - магазин	7,975	+0,3	+0,3	+0,9	+0,9				+0,3						11,93%	14,15%
5	Професионализм продавцов	6,485							+0,9	+0,9						1,91%	1,85%
6	Различные способы оплаты	6,264				+0,9	+0,9			+0,3						8,59%	8,03%
7	Удобное расположение магазинов	6,229									+0,3		+0,9			4,77%	4,43%
8	Расширенный график работы	4,98				+0,9			+0,9		+0,3			+0,9		6,44%	4,79%
9	Отношение к постоянным клиентам	4,113				+0,3									+0,9	7,16%	4,39%
10	Доставка до покупателя	4,036							+0,3		+0,9					4,77%	2,87%
11	Возможность предзаказа	4,006	+0,3	+0,3		+0,9										7,16%	4,28%
12	Реклама	3,354			+0,9	+0,9		+0,3			+0,1				+0,3	6,44%	3,22%
13	Сопутствующие товары	2,192	+0,9	+0,1												7,16%	2,34%
14	Доставка до машины	1,659														2,86%	0,71%
15	Дополнительная упаковка	0,892						+0,9								5,01%	0,67%
16	Дополнительная обработка товара	0,651										+0,9				6,44%	0,63%
Приоритет характеристики	Знак		+	+	+	+	+	+	+	+	+	+	+	+	+		
	Абсолютное занчение		38,8	24,93	9,044	28,61	4,486	0,657	8,072	8,956	6,179	0,586	4,454	4,482	5,861	100,00%	100,00%
	Относительное значение		26,73	17,18	6,232	19,72	3,091	0,453	5,563	6,172	4,258	0,404	3,069	3,088	4,039		

Поэтому показателя $K_i^{\text{отн}}$ в дальнейшем будет недостаточно, так как каждое требование покупателя имеет для него различный приоритет, поэтому вводится новый комплексный нормированный показатель V_i, который представляет собой:

$$V_i = \frac{P_i K_i^{\text{отн}}}{\sum_{i=1}^{i=N} P_i K_i^{\text{отн}}}, \tag{4}$$

где P_i – приоритет *i*-го требования потребителей.

Отметим, что показатель P характеризовал собой приоритет каждого отдельного требования, то, выраженный в процентах V_i представляет собой приоритет *i*-го требования, но уже не для потребителя, а для предприятия с учетом объективной ситуации на рынке на фоне конкуренции.

Если рассматривать «возведение» Дома качества как процесс, то на вход данного процесса подавались следующие данные:

– требования покупателей к продукции и предоставляемым услугам;

– приоритеты требований;

– данные сравнения предприятия с конкурентами в вопросе выполнения требований клиентов.

Тогда на выходе данного процесса получили следующую информацию:

– о более правильном распределении бюджета по статьям расхода с целью получения максимального результата при минимальных затратах;

– о требованиях покупателей к товарам и услугам в предоставлении которых превосходят конкуренты;

– о реальных возможностях для «прорывного» улучшения в организации работы данного предприятия по результатам бенчмаркинга.

Заключение

Данная работа посвящена использованию метода QFD в сфере торговли с целью развития бизнеса путем повышения клиентского спроса и расширения рынков сбыта. Рассмотрен глубокий анализ требований потребителя к продукции или услугам, на основании которого принимаются управленческие решения, направленные на увеличение клиентского спроса. В итоге метод QFD позволяет добиться максимального выполнения запросов потребителей при оптимизации расходов предприятия, что, в конечном итоге, будет способствовать развитию бизнеса.

Использование метода:

– позволяет определить требования потребителей, выделить среди них приоритетные и воплотить их в продаваемых товарах и предоставляемых услугах;

– обеспечивает гарантии того, что ваш товар или услуга будет востребован на конкретном сегменте рынка;

– обеспечивает вашему предприятию большую рыночную долю благодаря предложению товара, отвечающего всем запросам потребителя;

– позволяет оптимизировать бизнес-процессы предприятия на всех его стадиях;

– позволяет оптимально распределить, а значит – наиболее эффективно использовать ограниченные ресурсы предприятия;

– обеспечивает конкурентные преимущества предприятию за счет выполнения приоритетных требований клиента.

Метод QFD должен стать важным инструментом в руках грамотных менеджеров для работы над оптимизацией структуры и

деятельности предприятия с целью повышения клиентского спроса и расширения рынков сбыта.

Список литературы

1. Брагин, Ю В. Путь QFD: проектирование и производство продукции исходя из ожиданий потребителей / Ю. В. Брагин, В. Ф. Корольков. – Ярославль : Негосударственное некоммерческое образовательное учреждение «Центр качества», 2003. – 240 с.

2. Аронов, И. З. Стратегический маркетинг как основа обеспечения качества продукции / И. З. Аронов, Е. В. Подболотова // Методы менеджмента качества. – 1999. – № 4. – С. 15-26.

3. Корольков, В. Ф. Процессы управления организацией / В. Ф. Корольков, В. В. Брашн. – Ярославль : ред. изд. центр Яртелекома, 2001. – 415 с.

Реализация автоматической пропускной системы автотранспорта на основе многослойных нейронных сетей

А. С. Матяшов

Студент магистрант

Л. А. Минин

Доцент

Введение

На сегодняшний день нейронные сети решают все больший спектр задач, предоставляя новые подходы и инструменты решения. В данной работе предлагается рассмотреть вариант реализации системы распознавания автомобильных номерных знаков на основе многослойных нейронных сетей для последующей интеграции их с системой пропуска автотранспорта.

Ключевые слова: нейронные сети, python, персептрон, объектно-ориентированное программирование, безопасность.

Сегодня тема искусственного интеллекта весьма актуальна. Многие крупные IT-компании уже давно активно используют нейронные сети в своих проектах. Например, Yandex использует сверточные сети для распознавания и поиска похожих фотографий.

Одно из приложений машинного обучения и искусственного интеллекта – это распознавание образов. Оно может находить

применение в различных областях человеческой деятельности, в том числе при автоматизации систем контроля пропускного режима.

В данной работе хотелось бы поговорить о построение простейшей системы по распознаванию образов с применение клиент – серверной архитектуры. Основная цель – дать базовое представление о структуре подобного проекта, рассказать о нейронной сети, которая использовалась в данной работе, поговорить о микросервисной архитектуре. В результате, опираясь на данную работу, можно будет реализовать на практике подобное решение для автоматизации пропускной системы на предприятии.

1. Структура проекта

Ключевым элементом этой системы является нейронная сеть для распознавания отдельных символов — цифр и букв. Базовым элементом искусственной нейронной сети я является искусственный нейрон, представленный на рис. 1.

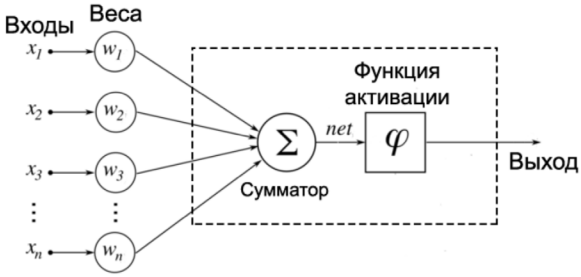


Рис. 1. Искусственный нейрон

Общая схема включает в клиентскую часть и сервер. Клиент расположен на удалённом объекте и может представлять из себя одноплатный компьютер, к которому подключается камера, датчики приближения объектов и выводы управления пропускной системы. Сервер, расположенный на виртуальной машине, нейронную сеть и REST API. Когда автомобиль подъезжает к воротам, датчик приближения инициирует событие, по которому камера делает снимок передней части авто. Далее полученный снимок через REST API отправляется на сервер, где нейронная сеть определяет автомобильный номер. Затем происходит поиск полученного номера в базе и, если таковой присутствует, сервер отправляет в виде ответа специальный json файл с разрешением на пропуск. Клиент, получив такой ответ от сервера, коммутирует контакты, питающие электрический замок. В

случае если в доступе отказано, владелец системы получает оповещение с севера о попытке доступа к объекту с дополнительной информацией и снимком с камеры.

Отличие многослойных нейронных сетей от однослойных заключается в том, что мы имеем не один, а несколько скрытых слоев помимо выходного слоя (все слои кроме последнего считаются скрытыми).

Способность искусственных нейронных сетей обучаться является их наиболее интригующим свойством. Для обучения сети в данной работе был выбран метод обратного распространения ошибки. Для реализации алгоритма был выбран язык Python [1]. Этот язык поддерживает объектно-ориентированное программирование [2], имеет простой синтаксис и существуют отличные среды для разработки, такие как PyCharm от компании JetBrains. Разработанное в данной работе программное обеспечение использует объектно-ориентированный парадигму программирования с применением известных паттернов проектирования [3]. В качестве системы контроля версий использовалась децентрализованная система Git и платформу GitHub в качестве сетевого репозитория.

2. Архитектура серверной части

Теперь следует рассказать о структуре серверной части. Первое - это пакет, который занимается обеспечением персистентности данных. Как уже говорилось ранее, общепринятой практикой является заполнение начальных весов нейронов случайными значениями. Но в таком состоянии каждого нейрона сеть бесполезна, так как она не обучена (под состоянием в данном случае я понимаю состоянием объектов класса Neuron). Сам процесс обучения может длиться достаточно долго, поэтому нецелесообразно каждый раз при необходимости какого-либо расчета заново обучать сеть. Отсюда можно сделать вывод, что практическое применение сеть имеет только тогда, когда она способна сохранять и загружать свое состояние. Класс для хранения весов — основная сущность, с которой взаимодействует нейрон через публичные методы. Грубо говоря, все, что «знает» нейрон об объекте этого класса, что он может либо сохранить его состояние, либо загрузить его.

После того, как сеть была обучена на наборе, состоящем из вырезанных цифр и букв из номерных знаков, пришло время написать некую прослойку между клиентом и нейронной сетью, которая позволит им взаимодействовать друг с другом посредствам HTTP запросов. Для её реализации использовалась концепцию REST API. В качестве

фреймворка можно использовать Flask — это достаточно простой инструмент для решения подобных задач.

Во время работы сервера могут возникать разные события. Например, ошибки, связанные с обработкой запроса, многократные попытки получить доступ к объекту, где установлена система и многое другое. Обо всех важных событиях, происходящих на сервере, хотелось бы получать в каком-либо виде нотификации. Для этих целей можно дополнительно написать специальный модуль.

Так как получившаяся реализация нейронной сети способная распознать только лишь отдельные символы, а с клиентской стороны приходит просто фотография авто, необходимо сделать предварительную обработку изображения – поиск границ номера и разбиение его на отдельные символы. Для этого можно использовать библиотеку OpenCV. В результате, получаем следующие сегменты отдельных изображений, представленных на рис. 2.

Рис. 2. Сегментирование изображения

3. Архитектура клиентской части

Клиентская часть программы намного проще, в ней отсутствует какая-либо обработка изображения и распознавание номерного знака. Вся эта работа ложится на плечи сервера. Сама реализация проста - при периодическом опросе датчиком приближения (могут быть установлены прямо в воротах) мы можем понять, что есть приближающийся объект, причем при наличии нескольких таких датчик мы автоматически исключаем ситуацию, когда клиент ошибочно отправляет запрос на сервер, например, при приближении домашних животных. Интеграция с уже имеющимися механизмами открывания тоже проста за счет использования силовых контакторов, даже если это электрические замки без какой-либо логики на борту.

Взаимодействие через незащищенный протокол HTTP несет в себе ряд ограничений и дыр в безопасности. Предполагается, что IP-адрес сервера известен и в этой ситуации без каких-либо механизмов защиты он уязвим к многим атакам. Проблема может быть решена путем

использования HTTPS-соединения. Общая схема взаимодействия клиента с сервером представлена на рис. 3.

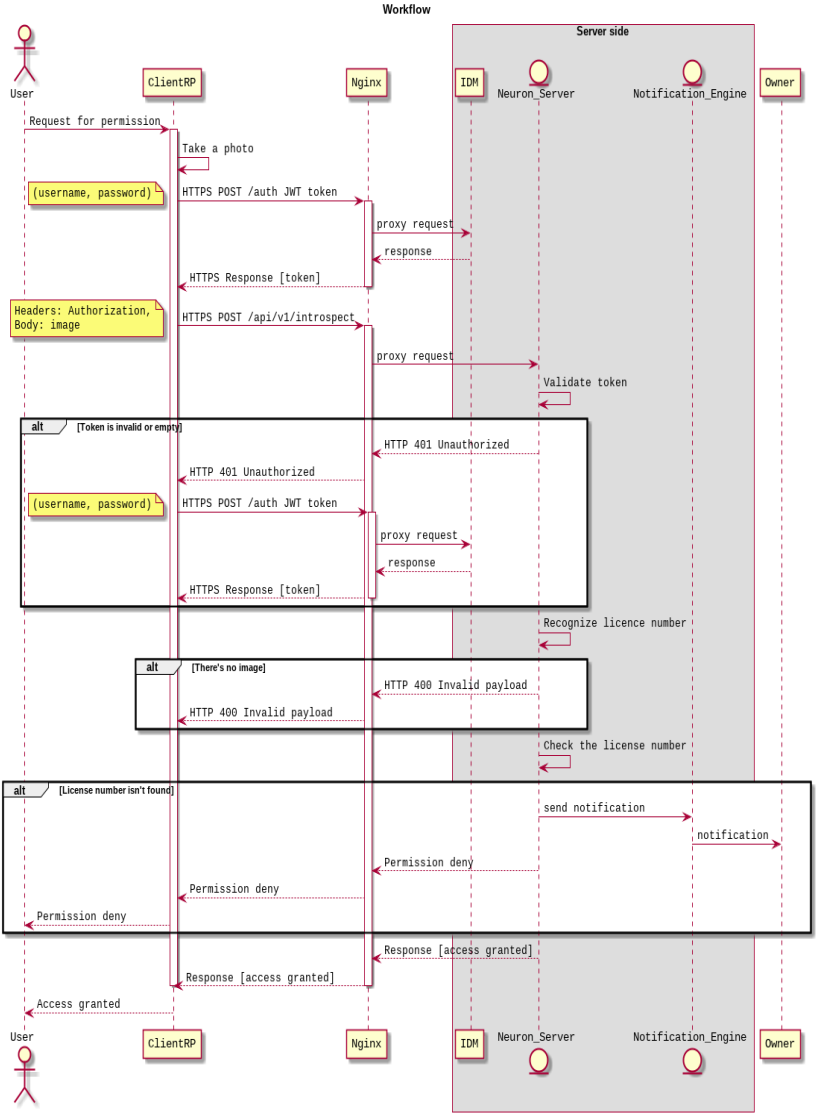


Рис. 3. UML диаграмма работы системы

Заключение

В данной статье был предложен алгоритм распознавания символов автомобильного номера и разработано программное обеспечение для системы пропуска автотранспорта на удалённых объектах. При разработке использовались технологии REST-запросов для общения между компонентами системы, контейнер для размещения сервера в облачных средах, технология JWT для обеспечения безопасности при общении клиента с сервером. Полученная система является более дешёвой по сравнению с имеющимися на рынке аналогами.

Модульная архитектура, заложенная на этапе проектирования системы позволяет легко расширять функционал. Можно ввести политики доступа, с разрешением въезда на объект только в определенное время, добавить интеграцию с внешними система аутентификации и авторизации или изменить каналы оповещения о событиях в системе.

Список литературы

1. Лутц, М. Изучаем Python / М. Лутц. – М. : Символ-Плюс, 2011. – 1280 с.
2. Фримен, Э. Паттерны проектирования / Э. Фримен. – Санкт-Петербург : Питер, 2003. – 656 с.
3. Вайсфельд, М. Объектно-ориентированное мышление / М. Вайсфельд. – СПб. : Питер, 2014. – 304 с.

Модель угроз безопасности информации на типовом рабочем месте

М. Ю. Модестов

Студент магистрант

С. А. Вялых

Доцент

Введение

В настоящее время владение информацией играет важную роль. У нарушителей может быть много мотиваций, нарушающих штатную обработку информации. Современные тенденции защиты информации и требования методических документов ФСТЭК России все больше делают акцент на анализ и устранение уязвимостей информационных

систем, включая использование методов тестирования на проникновение. Однако существующие методы анализа уязвимостей и тестирования на проникновение носят в основном узконаправленный характер, что обусловлено отсутствием системного подхода при анализе возможных ситуаций. Так банк данных угроз безопасности информации ФСТЭК России не учитывает жизненный цикл ИС и ее элементов и ограничен только этапом эксплуатации ИС. Анализ уязвимостей сводится, как правило, к использованию набора доступных средств автоматизации сканирования уязвимостей, а результаты тестирования на проникновение, определяя ряд способов преодоления защиты, не ставят целью полноту исследования всех возможных сценариев реализации угроз безопасности информации. Данная работа, основываясь на базовой модели угроз безопасности информации ФСТЭК России [1], преследует цель охватить все возможные сценарии реализации угроз безопасности информации на типовом рабочем месте пользователя с учетом жизненного цикла ИС и ее элементов для нарушителей имеющих низкий потенциал.

Модель угроз безопасности информации на типовом рабочем месте

Для составления модели необходимо правильно понимать какие существуют угрозы безопасности и возможные уязвимости для отдельного компьютера. Угрозы могут быть реализованы на различных этапах жизненного цикла компьютера и/или его комплектующих. На рис. 1 представлена модель угроз безопасности для отдельного компьютера на этапах жизненного цикла.

Данная модель основывается на «Базовой модели угроз» [1] в отличие от которой основной акцент сделан на наиболее просто реализуемых способах реализации угроз безопасности информации в информационной системе в зависимости от времени начала их реализации и возможностей удаленного (сетевого) или физического доступа к атакуемому компьютеру или его элементам используемых на рабочем месте. В модели были рассмотрены следующие основные этапы: до эксплуатации, эксплуатация и пост-эксплуатация. Далее разберем каждый этап жизненного цикла подробнее.

До эксплуатации компьютера относится разработка и дистрибуция программного обеспечения и комплектующих компьютера. К разработке относится создание программного обеспечения и всех комплектующих. К дистрибуции относится, как сборка компьютера, так и установка ОС, драйверов и программного обеспечения на него. При сборке существует уязвимость установки аппаратной закладки в компьютер, или уже перепрограммированного оборудования, такого как видеокарта, сетевой адаптер или других внутренних устройств.

Рис. 1. Модель угроз безопасности
на этапах жизненного цикла компьютера

Даже изменения в BIOS могут оказать вред при дальнейшем использовании компьютера. При установке программного обеспечения не исключается установка дополнительного вредоносного программного обеспечения, драйверов или файлов, о которых последующий пользователь не будет догадываться.

Следует отметить, что современные нормативные документы, в частности 17 и 21 приказы ФСТЭК [2-3], не отменяя «Базовой модели угроз» [1], обходят молчанием угрозы, связанные с закладками, с заранее внедренными вредоносными программами.

Внедренные в самом раннем этапе жизненного цикла уязвимости в последующей работе компьютера будут сложнее обнаруживаться и могут нанести значительный вред.

При эксплуатации компьютера, когда он уже находится на типовом рабочем месте, нарушитель может получить к нему физический или сетевой доступ.

При физическом доступе возможны 2 сценария реализации угроз безопасности информации: получение доступа к выключенному компьютеру и к включенному, который уже авторизован в системе.

Когда компьютер выключен, нарушитель может установить аппаратную закладку, залезть в BIOS, попытаться перепрограммировать микропрограммы видеокарты, сетевого адаптера или других внутренних устройств. Нарушитель может с загрузочного носителя запустить другую систему (дистрибутив Linux) и после изменить файлы в основной системе, загрузить какое-либо вредоносное программное обеспечение, скопировать системные файлы, содержащие информацию о паролях или каким-либо другим способом нарушить работу компьютера в основной системе.

Когда компьютер включен и авторизован в системе нарушитель, может внести изменения в работу системы, скомпрометировать пароль, установить вредоносные программы, повысить привилегия, обойти и деструктивно воздействовать на СЗИ.

Находясь в одной сети, можно провести ее сканирование с помощью сканеров уязвимостей, после чего нарушитель может попытаться ими воспользоваться. Получив первичный доступ к компьютеру удаленно, возможны все те же действия, что и при физическом доступе компьютера, авторизованного в системе, а также может быть перехвачена или изменена информация исходящая или приходящая к компьютеру или атакованы пароли удаленных сервисов, что может дать нарушителю дальнейшую возможность подключения к другим устройствам в системе.

Нельзя исключать уязвимости нулевого дня, когда уязвимость или атака становится публично известной, а производители ПО еще не выпустили исправления.

Также существуют угрозы при серфинге в интернете пользователя компьютера. Посещение неблагонадежных сайтов или подмена на сайте какого-либо содержимого, что может привести к последующему заражению компьютера. Такие угрозы могут быть обусловлены слабостями механизмов фильтрации сетевого трафика и антивирусного контроля на уровне организации.

Существуют угрозы компрометации взаимодействующих и вывода из строя обеспечивающих систем, например, компрометация электронной подписи или обновления ПО.

Во время эксплуатации может возникнуть такая ситуация, как ремонт. К компьютеру будет предоставлен физический доступ, а также возможно извлечение, удаление конфиденциальной информации, которая осталась на жестких дисках или хищение средств хранения, обработки.

В пост-эксплуатационном периоде, когда компьютер уже не используется на типовом рабочем месте, на нем может остаться какая-либо информация и поэтому, как и при ремонте, существует угроза извлечение конфиденциальной информации, которая осталась на жестких дисках.

Никто не исключает человеческий фактор. Ошибки и действия людей могут произойти на любом этапе жизненного цикла. Они могут быть случайными или намеренными. То могут быть методы социальной инженерии, фишинга, преднамеренного нарушения установленных требований, утрата носителей информации или какая-либо компрометация конфиденциальной информации, которая могла произойти случайно.

Угрозы были сопоставлены с банком данных угроз (БДУ) безопасности информации ФСТЭК [4]. В БДУ на данный момент содержится 217 возможных угроз. Из них рассматривались угрозы, которые могут произвести нарушители с низким потенциалом. Низкий потенциал подразумевает наличие возможностей уровня одного человека по приобретению (в свободном доступе на бесплатной или платной основе) и использованию специальных средств эксплуатации уязвимостей.

Заключение

Данная работа, основываясь на базовой модели угроз безопасности информации ФСТЭК России [1], преследует цель охватить все возможные сценарии реализации угроз безопасности информации на

типовом рабочем месте пользователя с учетом жизненный цикла ИС и ее элементов и для нарушителей имеющих низкий потенциал. Предлагаемый подход позволяет использовать разработанную модель при оценке реальной защищенности информационной системы и проведения ее тестирования на проникновение с учетом действий людей, возможных источников угроз безопасности информации, а также жизненного цикла средств вычислительной техники.

Список литературы

1. Базовая модель угроз безопасности персональных данных при их обработке в информационных системах персональных данных [Электронный ресурс] : ФСТЭК России. – Режим доступа : https://fstec.ru/component/attachments/download/289
2. Приказ ФСТЭК России №17 «Об утверждении Требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах» [Электронный ресурс] : ФСТЭК России. – Режим доступа : https://fstec.ru/component/attachments/download/566
3. Приказ ФСТЭК России №21 «Об утверждении состава и содержания организационных технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных» [Электронный ресурс] : ФСТЭК России. – Режим доступа : https://fstec.ru/component/attachments/download/561
4. Банк данных угроз безопасности информации [Электронный ресурс] : ФСТЭК России. – Режим доступа : https://bdu.fstec.ru/threat

Система формирования квантовой логики при реализации алгоритма Шора на многокубитовых регистрах

Н. А. Насонова

Студент магистрант

С. А. Запрягаев

Профессор

Введение

В настоящее время квантовые информационные системы являются одними из наиболее интенсивно развивающихся направлений научных

исследований, это обусловлено практическим использованием и применением квантового компьютера.

Квантовые технологии объединяют следующие основные сферы: квантовые вычисления, квантовые коммуникации, квантовую метрологию и сенсорику. Квантовые компьютеры и квантовые симуляторы могут решать вычислительные задачи, «неподъемные» для классических суперкомпьютеров.

За последние 30 лет появились реализуемые на квантовом компьютере алгоритмы, которые гораздо эффективнее классических. Это касается, например, задачи факторизации – поиска простых сомножителей, из которых состоит число. В случае достаточно больших чисел такая задача крайне трудна для классических компьютеров, но потенциально может быть реализована на квантовых компьютерах с помощью алгоритма, предложенного Шором. Практический интерес к ней связан с возможностью взлома ряда алгоритмов криптографии с открытым ключом, используемых сегодня повсеместно.

1. Алгоритм Шора

Алгоритм Шора [1] – квантовый алгоритм факторизации (разложения числа на простые множители), позволяющий разложить число N за время $O(\log^3 N)$, используя $O(\log N)$ логических кубит.

Алгоритм Шора [1], используя возможности квантовых компьютеров, способен произвести факторизацию числа не просто за полиномиальное время, а за время, не намного превосходящее время умножения целых чисел (то есть практически так же быстро, как происходит само шифрование).

Квантовый алгоритм Шора [1-3] на рис. 1 использует два квантовых регистра – регистр данных $|k>_n$ и регистр результата вычисления функции $|y>_n$, первоначально находящихся в нулевом базисном состоянии.

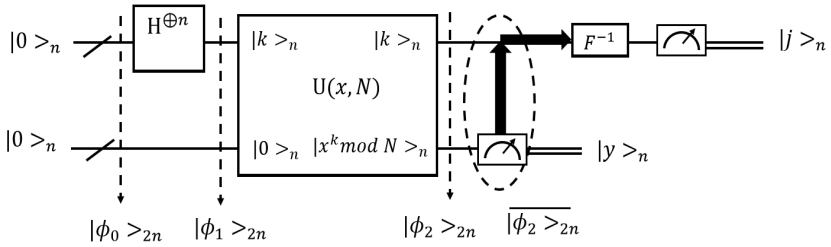


Рис. 1. Схема квантового алгоритма Шора

Первый шаг квантового алгоритма состоит в переводе начального состояния $|0>_n$, регистра $|k>_n$ в равновероятную суперпозицию всех базисных состояний регистра из n кубит путём применения операции Уолша-Адамара. Регистр $|y>_n$, на данном этапе не меняется.

Допустим далее, что имеется некоторый оператор (квантовый гейт) U_f, создающий в регистре y вычисленное значение функции

$$f(x) = x^2 \bmod N, \text{ где } x < N. \quad (1)$$

Числа x, N являются целыми и взаимно простыми, для которых $\gcd(x, N) = 1$. Тогда вторым шагом квантовой части алгоритма является преобразование вида:

$$U_f : |\vartheta_1>_{n+n} \to |\vartheta_2>_{n+n}. \quad (2)$$

формирующее новое состояние $|\vartheta_2>_{n+n}$, в этом случае есть:

$$|\vartheta_2>_{n+n} = \sqrt{\frac{1}{2^n}} \sum_{k=0}^{2^n-1} |k>_n \otimes |x^k \bmod N>_n. \quad (3)$$

Измерение регистра Y является третьим шагом алгоритма Шора. В результате выполнения которого в регистре формируется суперпозиция регистров, определяемых набором десятичных чисел, имеющих период, равный периоду r.

Далее применяется обратное Фурье преобразование и измеряется регистр x. После всех преобразований представленных на схеме возникает простой алгоритм определения периода числа по модулю r.

2. Квантовый компьютер IBM

Целью работы было построение многокубитовой логики при реализации квантовой машины алгоритма Шора, построение и тестирование схем в квантовом компьютере IBM и анализ полученных результатов.

Пользователи IBM компьютера взаимодействуют с квантовым процессором через квантовую схему вычислений, применяя квантовые операторы к кубитам, используя графический интерфейс, называемый квантовым композитором или через язык Qiskit. как показано на рис. 2.

Схемы строятся с помощью набора квантовых гейтов, как показано на рис. 3. Далее пользователю необходимо запустить программу и проанализировать результат.

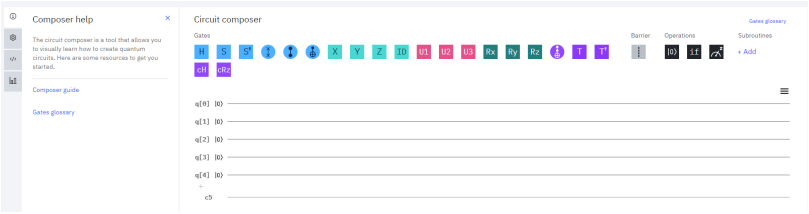


Рис. 2. Набор квантовых гейтов в Composer

Рис. 3. Построение схем в Composer

3. Построение квантовой информационной системы при реализации алгоритма Шора в квантовом компьютере IBM

Была разработана квантовая информационная система для случая, $(x=2, N=15)$ исходный регистр $< x >< f(x) >$ $|x_1, x_2, x_3, x_4, x_5; y_1, y_2, y_3, y_4, y_5>$, как показано на рис. 4.

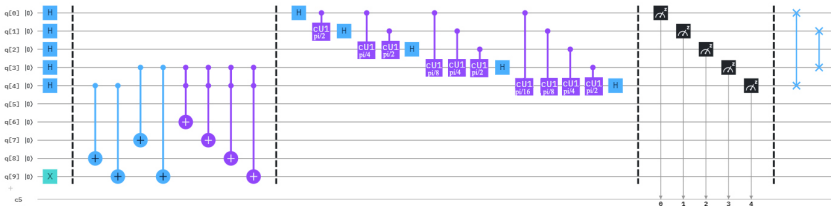


Рис. 4. Квантовая информационная система в IBM Q

Построение состоит из нескольких этапов:

1. Объявление регистров кубит. Применение гейта Адамара к кубитам $q[0]-q[4]$. Применение гейта X к $q[9]$ кубиту и барьер, применимый ко всем кубитам.

2. Построение определенного набора последовательности, гейтов контролируемого и дважды контролируемого отрицания. Набор гейтов для разных схем различен. После применяется гейт барьер для предотвращения смешивания кубитов во времени.

3. Десятикубитовый регистр после выполнения последнего гейта на схеме примет вид: (где C_i – амплитуды базисных состояний кубита).

$$\begin{aligned}
&|C_0;C_1^{'}>+C_1;C_2^{'}>+C_2;C_4^{'}>+C_3;C_8^{'}>+\\
&|C_4;C_1^{'}>+C_5;C_2^{'}>+C_6;C_4^{'}>+C_7;C_8^{'}>+\\
&|C_8;C_1^{'}>+C_9;C_2^{'}>+C_{10};C_4^{'}>+C_{11};C_8^{'}>+\\
&|C_{12};C_1^{'}>+C_{13};C_2^{'}>+C_{14};C_4^{'}>+C_{15};C_8^{'}>+\\
&|C_{16};C_1^{'}>+C_{17};C_2^{'}>+C_{18};C_4^{'}>+C_{19};C_8^{'}>+\\
&|C_{20};C_1^{'}>+C_{21};C_2^{'}>+C_{22};C_4^{'}>+C_{23};C_8^{'}>+\\
&|C_{24};C_1^{'}>+C_{25};C_2^{'}>+C_{26};C_4^{'}>+C_{27};C_8^{'}>+\\
&|C_{28};C_1^{'}>+C_{29};C_2^{'}>+C_{30};C_4^{'}>+C_{31};C_8^{'}>.
\end{aligned} \tag{4}$$

4. Обратное Фурье образование, измерение регистра кубит и гейт SWAP.

Набор последовательности гейтов, построенных на шаге 2, представлен на рис. 5 в виде кода на языке QASM:

```
cu1(pi/2) q[0],q[1];
h q[1];
cu1(pi/4) q[0],q[2];
cu1(pi/2) q[1],q[2];
h q[2];
cu1(pi/8) q[0],q[3];
cu1(pi/4) q[1],q[3];
cu1(pi/2) q[2],q[3];
h q[3];
cu1(pi/16) q[0],q[4];
cu1(pi/8) q[1],q[4];
cu1(pi/4) q[2],q[4];
cu1(pi/2) q[3],q[4];
h q[4];
barrier q[0],q[1],q[2],q[3],q[4]
measure q[0] -> c[0];
measure q[1] -> c[1];
measure q[2] -> c[2];
measure q[3] -> c[3];
measure q[4] -> c[4];
barrier q[0],q[1],q[2],q[3],q[4]
swap q[0],q[4];
swap q[1],q[3];
```

Рис. 5. Код QASM

4. Результат запуска квантовой информационной системы и анализ результата

В результате исполнения алгоритма получим числа (рис. 6). Для вычисления разложения числа на простые множители подойдет число 16. Так как $r = 16, k / r = 16 / 24 = 4 / 6.$ То есть период равен 4.

Таким образом, $C_1 = 2^2 + 1 = 5$ и $C_2 = 2^2 - 1 = 3,$ $C_1 \times C_2 = 15.$

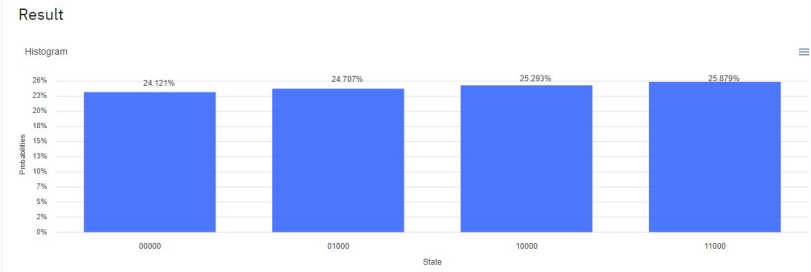


Рис. 6. Результат работы алгоритма

5. Заключение

Данная статья посвящена разработке и реализации схем при реализации алгоритма Шора на многокубитовых регистрах. В ходе работы были изучены принципы построения квантовых компьютеров для реализации алгоритма Шора, построение моделей схем в IBM Q Experience. Был изучен язык QASM для построения моделей в облачном квантовом компьютере. Были сделаны следующие выводы:

– квантовый компьютер – это сложная система, которую невозможно смоделировать с помощью классических вычислительных технологий, поскольку сложность растет экспоненциально с увеличением размера системы;

– сформулированный подход позволяет спроектировать последовательную систему квантовых операторов для вычисления конкретной квантовой функции заданной квантовой машины. Конкретная реализация квантового компьютера для реализации алгоритма Шора зависит от выбранных параметров, заданного числа N и x для вычисления функции возведения числа в степень по модулю;

– квантовый компьютер не является стандартным и не может быть представлен в универсальном виде, для каждой определенной схемы требуется уникальный подход составления алгоритма работы квантовой машины.

Список литературы

1. Запрягаев, С. А. Введение в квантовые информационные системы : учеб. пособие / С. А. Запрягаев. – Воронеж : изд. дом Воронежского государственного университета, 2015. – 219 с.
2. Давыдов, А. С. Квантовая механика / А. С. Давыдов. – М. : Наука, 1973. – 439 с.
3. Ожигов, Ю. И. Квантовые вычисления / Ю. И Ожигов. – М. : Макс Пресс, 2003. – 152 с.

Разработка виртуального окружения для тестирования устройств интернета вещей, базирующихся на IP сетях

В. А. Новиков

Студент магистрант

Д. И. Соломатин

Старший преподаватель

Введение

В настоящее время разрабатывается большое число устройств интернета вещей. Такие устройства находят применение в различных сферах жизни: домашняя и промышленная автоматизация, транспорт, медицина. Для подобного рода устройств особое внимание следует уделить вопросу тестирования, поскольку многие из них могут физически взаимодействовать с окружающим миром – по ошибке включенная розетка может даже привести к пожару.

Очевидно, физическое устройство невозможно протестировать исключительно программными методами. Однако, определенный набор функциональности может быть автоматизировано протестирован.

По определению, устройства интернета вещей не являются изолированными, но взаимодействуют с другими устройствами в рамках некоторой сети. Возникает проблема интероперабельности – устройства (скорее всего, созданные разными производителями) должны корректно взаимодействовать друг с другом в рамках заранее определенного протокола.

Существует множество протоколов интернета вещей. Многие из них базируются на «нестандартных» сетевых технологиях (ZWave,

Zigbee) и с трудом уживаются с окружающим миром, который, по большей части, использует TCP/IP.

В рамках данной работы рассматривается подход к автоматизированному тестированию устройств интернета вещей, базирующихся на IP сетях. Детально будет рассмотрен протокол Dotdot [1], однако все изложенные результаты могут быть адаптированы к любому основанному на IP протоколу.

1. Основные понятия Dotdot

Zigbee Alliance была выпущена спецификация Dotdot, описывающая протокол взаимодействия для устройств интернета вещей. Dotdot основывается на других спецификациях и протоколах:

- CoAP (RFC 7252) – транспортный протокол, работающий на основе UDP [2]. Dotdot использует данный протокол для передачи сообщений.
- ZCL – спецификация, описывающая модель данных для устройств интернета вещей – структуру сообщений, с помощью которых IoT устройства взаимодействуют друг с другом, абстрагировано от способа их сериализации и передачи [3].
- CBOR (RFC 7049) – формат сериализации данных. Dotdot использует данный формат для сериализации сообщений, структура которых определяется спецификацией ZCL [4].

Сама спецификация Dotdot описывает способ передачи сообщений ZCL в IP сетях, при помощи протокола CoAP.

ZCL (Zigbee Cluster Library) вводит понятие “кластер” [5] – набор атрибутов и команд, описывающий некоторую функциональность. Наиболее простой пример – OnOff кластер. Он состоит из:

- команд: On, Off, Toggle;
- одного атрибута: state – текущее состояние, включено или выключено.

Любое устройство, которое может включаться и выключатся, реализует этот кластер. Пример – лампочка, термостат, и т. д. Данный кластер абстрактен – эффект включения и выключения зависит от самого устройства.

ZCL определяет множество различных кластеров (отсюда и название – библиотека кластеров). Функциональность конкретного устройства составляется из некоторого подмножества кластеров, каждый из которых предоставляет стандартный API для работы с той или иной частью функциональности. При этом спецификация позволяет пользователям определять собственные кластеры, если некоторая функциональность не реализуется каким-либо из стандартных кластеров.

Множество экземпляров кластеров может быть объединено в «эндпоинт». Эндпоинт – это конкретное приложение, функциональность которого описывается множеством его кластеров. В рамках одного физически адресуемого устройства (имеющего конкретный IP адрес) может существовать несколько логических устройств, каждый из которых представлен своим эндпоинтом.

Отношение понятий ZCL представлено на рис. 1.

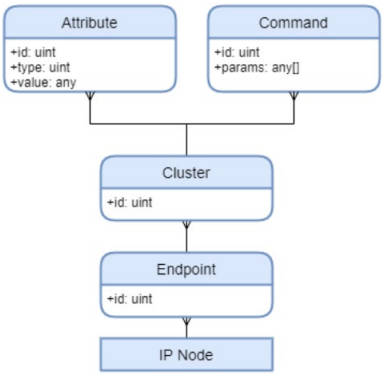


Рис. 1. Основные сущности ZCL

ZCL определяет модель данных, тогда как Dotdot определяет способ их передачи в IP сети. Dotdot задает, с помощью каких CoAP команд осуществляется работа с атрибутами и командами устройства. Например, чтобы прочитать атрибут №0 кластера №1 на эндпоинте №2 у устройства с адресом ff03::fd, нужно послать следующий CoAP GET–Request: coap://[ff03::fd]//zcl/e/2/s1/a?f=0

Dotdot также определяет формат ответа, который должен быть сгенерирован как результат выполнения данного запроса.

2. Реализация системы

В рамках данной работы был создан инструмент, позволяющий осуществлять интеграционное тестирование Dotdot устройств в автоматическом режиме.

Для тестирования используются следующие подходы:

– Тестирование осуществляется в рамках виртуальной изолированной IP сети. Тестируемое устройство работает в рамках этой сети.
– Тестирование осуществляется при помощи эталонного устройства, которое некоторым образом (согласно тестовой

процедуре) взаимодействует с тестируемым устройством в рамках виртуальной сети.
– Тестирование основано на анализе сетевого трафика между эталонным устройством и тестируемым устройством.

Тестируемое устройство не обязательно должно быть в единственном экземпляре – в рамках виртуальной сети может быть развернуто некоторое множество тестируемых устройств, взаимодействующих друг с другом и с эталонными устройствами.

В рамках разработки данного инструмента созданы:

– Эталонная реализация спецификации Dotdot. На основе данной реализации создается описанное выше эталонное устройство. В дальнейшем эталонную реализацию будем называть Test Harness, а тестируемое устройство – DUT (Device Under Test).
– Тестовая система, обеспечивающая работу с виртуальной сетью, исполнение тестовых сценариев, захват трафика в сети. Тестовый сценарий представляет собой скрипт, описывающий правила поведения для Test Harness (необходимое согласно тестовой процедуре) и правила анализа сетевого трафика.

Схематично процесс тестирования показан на рис. 2.

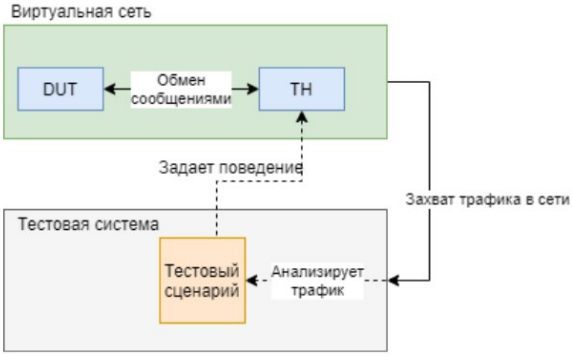


Рис. 2. Процесс тестирования с помощью разработанной системы

Test Harness представляет собой программную реализацию Dotdot устройства, поведение которого может быть динамически задано при помощи сценариев на языке Python.

Приложение состоит из следующих компонентов:

– Ядро, реализующее базовые механизмы протокола Dotdot: чтение и запись атрибутов, отправка команд. Также в рамках ядра реализовано некоторое подмножество библиотеки

кластеров. Ядро реализовано на языке C. В качестве реализации протокола CoAP используется библиотека libcoap. Для работы с данными в формате CBOR используется библиотека libcbor.

– Прослойка на языке Python, предоставляющая удобный для прототипирования API. С использованием данного API реализуются сценарии, описывающие поведение устройств.
– Поведенческий сценарий.

Структура Test Harness схематично представлена на рис. 3.

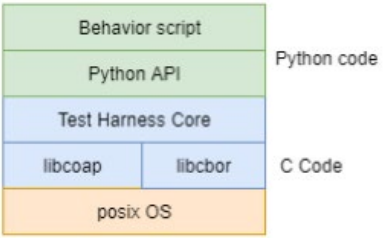


Рис. 3. Структура Test Harness

В листинге 1 представлен пример сценария для создания устройства–выключателя (OnOff Server кластер).

Листинг 1

Пример поведенческого скрипта для реализации простейшего устройства с OnOff кластером

```
class DemoOnOffSwitch(Endpoint):
    def __init__(self):
        self._onoff = OnOffServer()

        self.add_cluster(self._onoff)

    @dotlike.action(«set_state»)
    def set_state(self, address: address, state: bool):
        if state:
            self._onoff.send_on(address)
        else:
            self._onoff.send_off(address)

    @dotlike.action(«switch_state»)
    def switch_state(self, address: address):
        self._onoff.send_switch(address)

class DemoOnOffSwitchDevice(Device):
    def __init__(self):
        self._switch = DemoOnOffSwitch()
        self.add_endpoint(id=1, self._switch)
```

Класс DemoOnOffSwitch описывает эндпоинт с одним кластером – OnOffServer.

Класс DemoOnOffSwitchDevice описывает устройство в целом, которое, в данном случае, состоит из одного эндпоинта с номером 1.

Класс OnOffServer является оберткой над реализацией соответствующего кластера в ядре.

Методы, помеченные декоратором dotlike.action, позволяют управлять приложением извне: для этого необходимо отправить CoAP POST запрос на адрес coap://[address]/dottest/[endpoint_id]/[action], где address – сетевой адрес устройства, endpoint_id и action – номер эндпоинта и имя действия, которые необходимо выполнить. В теле сообщения можно передать параметры: они должны быть представлены в виде словаря в формате CBOR.

Например, чтобы выполнить действие set_state, необходимо отправить запрос, представленный в листинге 2.

Листинг 2

CoAP запрос для выполнения действия set_state

```
coap://[ff03::fd]//dottest/1/set_state POST
{«address»: «ff03::fe», state: True}
```

Как результат, устройство отправит команду On на устройство с адресом ff03::fe.

Тестовая система состоит из двух подсистем:

1. Backend: Тестовый движок. Работает как демон, отвечает за управление виртуальными сетями, за запуск и анализ тестов. Написан на языке Python. Предоставляет json-rpc [6] API для управления.
2. Frontend: предоставляет графический интерфейс для работы с системой. Взаимодействует с тестовым движком через json-rpc. Реализован на языке Python с использованием библиотеки PySide2 [7]

Архитектура тестовой системы схематично представлена на рис. 4.

Тестовая система может работать удаленно – например, на тестовом стенде, к которому подключены реальные устройства. По этой причине графический интерфейс отделен от тестового движка.

Для создания виртуальной сети, в рамках которой осуществляется тестирование, используется docker. И тестируемое устройство, и Test Harness запускаются как docker контейнеры в рамках виртуальной (bridge) сети. Тестовый движок позволяет добавлять, просматривать и удалять виртуальные сети, запускать и останавливать контейнеры.

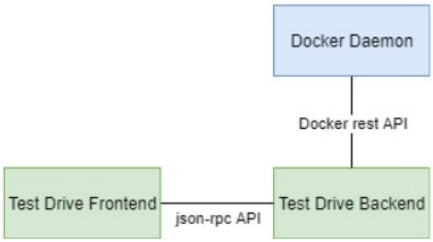


Рис. 4. Архитектура тестовой системы

Для наблюдения за сетью используется tshark. Tshark является консольной версией Wireshark – программы для захвата и анализа сетевого трафика. Tshark содержит описания большинства существующих сетевых протоколов, и возвращает декодированные сообщения в формате XML, удобные для дальнейшего анализа.

Процесс интеграционного тестирования можно кратко описать следующим образом:

1. Формируется сеть: запускается тестируемое и тестирующие устройства.
2. Тестирующее устройство посылает определенные команды на тестируемое устройство.
3. Тестируемое устройство генерирует ответ.
4. Ответ анализируется тестовой системой.

Тест представляет собой Python скрипт, написанный с использованием библиотеки pytest [8]. Пример тестовой процедуры представлен в листинге 3.

Листинг 3

Пример тестовой процедуры

```
def test_send_on(test_harness, dut, runner):
    test_harness.load_from(«demo_on_off_switch.py»)


    runner.info(«sending on command...»)
    test_harness.set_state(dut.address, state=True)
    resp = dut.wait_for(DefaultResponse, timeout=3)
    runner.info(«response received»)

    assert resp.status == Status.SUCCESS
```

В данном примере тестовая процедура принимает три аргумента (фикстуры):

1. Test Harness – объект для взаимодействия с экземпляром тестирующего устройства. Создается динамически на время

работы теста. Таких аргументов может быть несколько – все аргументы, имя которых начинается с test_harness, создают собственный экземпляр Test Harness. С помощью этого объекта задается поведение теста – отправка тех или иных команд на тестируемое устройство.

2. Dut (Device under test) – Объект, описывающий тестируемое устройство. Позволяет получить доступ к потоку сообщений (от данного устройства), захваченных в виртуальной сети. В дальнейшем захваченные сообщения анализируются тестовой процедурой.
3. Runner – объект для взаимодействия с интерфейсом системы. Позволяет вывести сообщение в пользовательский интерфейс или запросить ввод от пользователя.

Запуск теста состоит из следующих шагов:

1. Пользователь (это может быть как человек, работающий с GUI системы, либо программа, взаимодействующая с движком по json–rpc) подготавливает docker контейнер с тестируемым приложением и регистрирует его в тестовом движке.
2. Пользователь выбирает тестируемое приложение (из списка ранее зарегистрированных) и запускает его в некоторой виртуальной сети.
3. Пользователь выбирает тестовый скрипт для запуска и выбирает устройство (из списка ранее запущенных), которое будет тестируемым в данном запуске.
4. Тестовый движок создает экземпляры Test Harness, необходимые для запуска тестовой процедуры – запускает соответствующее число контейнеров с приложением Test Harness.
5. Тестовый движок выполняет тестовую процедуру.

Тестовая процедура задает поведение Test Harness при помощи поведенческого скрипта (функция load_from загружает скрипт в само приложение Test Harness. В данном примере используется поведенческий скрипт, описанный выше), затем генерирует некоторое количество сообщений, согласно тестовой спецификации. Затем, на основе анализа потока сообщений от тестируемого устройства, принимается решение об успешности тестового запуска.

В описанном примере тестовая функция генерирует команду On (при помощи действия set_state, описанного в поведенческом скрипте), а затем ожидает ответ (сообщение типа DefaultResponse) от тестируемого устройства и проверяет, что ответ содержит статус SUCCESS – т. е. команда была успешно выполнена.

На рис. 5 представлено основное окно графического интерфейса системы. В нижней части окна отображается список установленных docker образов и поле для вывода информационных сообщений. В верхней части окна отображается список запущенных образов, а также информация о захваченных в виртуальной сети сообщениях.

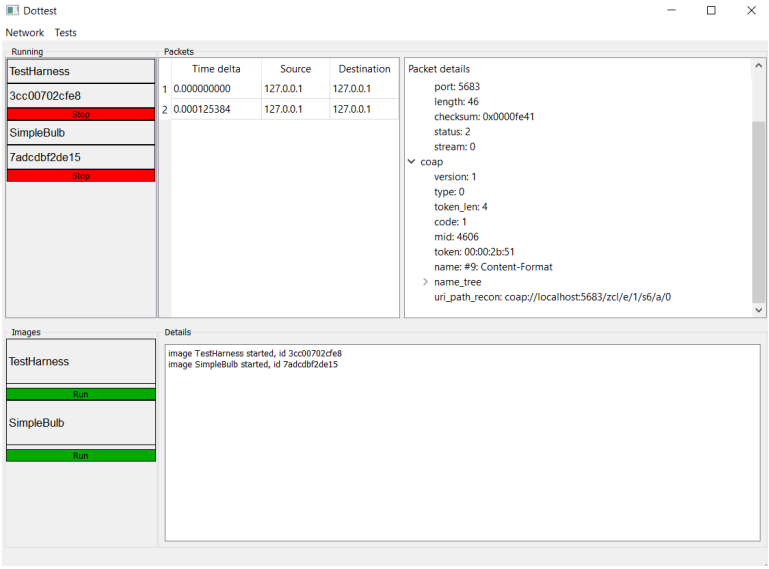


Рис. 5. Основное окно графического интерфейса системы

3. Примеры использования системы

Допустим, некоторая компания выпустила свою реализацию некоторого IoT устройства – например, управляемой розетки. Компания хочет протестировать реализацию на соответствие спецификации Dotdot. Тестирование возможно в двух вариантах: полностью в виртуальной среде, или на физическом устройстве.

Для первого варианта разработчикам необходимо представить приложение розетки в виде docker контейнера. Для этого приложение необходимо подготовить к запуску в рамках ОС:

– заменить функции, работающие с аппаратным обеспечением (например, управление реле для включения\выключения розетки, и т. д.) на функции–заглушки;
– сделать слой работы с сетью совместимым с posix–сокетами, поскольку именно такой сетевой интерфейс доступен для приложений, запущенных в docker контейнере.

Второй пункт, как правило, выполнен – поскольку многие ОС для встроенных систем реализуют posix сокеты. После описанной подготовки приложение упаковывается в docker контейнер и устанавливается в систему тестирования.

Допустим, разработчики тестируют функциональность OnOff кластера розетки. Для этого им необходимо:

– Подготовить поведенческий скрипт для Test Harness, который будет посылать розетке соответствующие команды. Скрипт, в целом, будет похож на таковой из листинга 1.
– Подготовить тестовый скрипт, который проанализирует трафик между Test Harness и розеткой. В простейшем случае скрипт будет выглядеть как таковой из листинга 3.
– Запустить подготовленный ранее docker контейнер и тестовый скрипт, дождаться результатов тестирования.

Тестовая система позволяет проверить соответствие поведения розетки спецификации, в данном случае:

– розетка на команду On отправила корректный ответ;
– розетка в ответ на команду чтения атрибута состояния отвечает значением True.

Однако, соответствие спецификации не означает, что розетка действительно включилась. Данную проверку нельзя формализовать – побочный эффект от «включения» чего-либо может быть разный.

Для полного тестирования возможен второй вариант – подключение физического устройства в виртуальную сеть, и его тестирование с помощью Test Harness. В этом случае разработчикам необходимо запустить программу на аппаратной платформе розетки, и с помощью соответствующего моста подключить розетку в виртуальную сеть.

Допустим, розетка использует WiFi для подключения к сети – в этом случае можно воспользоваться технологией Software Access Point [9]. WiFi точка доступа, поддерживающая данную технологию, подключается к docker bridge сети. При этом любое устройство, подключенное к данной точке доступа, попадает в виртуальную сеть. Схематично это представлено на рис. 6.

Для физического тестирования можно использовать те же поведенческие и тестовые скрипты, как и для виртуального тестирования. Разница лишь в том, что тестируемое устройство представлено не в виде контейнера, а в виде физического устройства (что, на самом деле, безразлично с точки зрения анализа сетевого трафика – на чем и построена тестовая система).

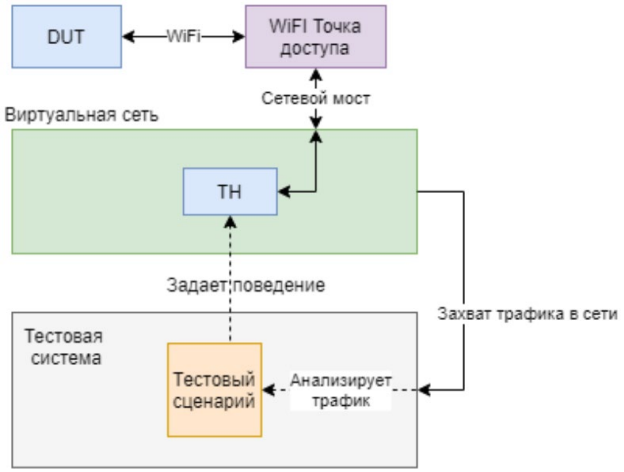


Рис. 6. Технология Software Access Point

После запуска тестового скрипта и анализа результатов, тестировщику остается проверить напряжение на выводах розетки, т. е. проверить, что розетка не только корректно (согласно спецификации) отвечает на команду включения, но и на самом деле включается.

Заключение

В данной работе был рассмотрен подход к автоматизированному интеграционному тестированию устройств, созданных на основе спецификации Dotdot. Был разработан инструмент, позволяющий осуществить такого рода тестирование. Разработанный инструмент абстрагирован от протокола взаимодействия, и может быть легко адаптирован к любому основанному на IP протоколу.

Список литературы

1. Dotdot over IP Base Device Behavior Specification [Электронный ресурс] : база данных. – Режим доступа : https://zigbeealliance.org/wp–content/uploads/zip/dotdot–ip–package.zip

2. The Constrained Application Protocol (CoAP) [Электронный ресурс] : база данных. – Режим доступа : https://tools.ietf.org/html/rfc7252

3. ZigBee Cluster Library Specification [Электронный ресурс] : база данных. – Режим доступа : https://zigbeealliance.org/wp–content/uploads/zip/zigbee–cluster–library.zip

4. Concise Binary Object Representation (CBOR) [Электронный ресурс] : база данных. – Режим доступа : https://tools.ietf.org/html/rfc7049

5. Gislason, D. Zigbee Wireless Networking / D. Gislason. – New York: Newnes, 2008. – 448 c.
6. JSON–RPC 2.0 Specification [Электронный ресурс] : база данных. – Режим доступа : https://www.jsonrpc.org/specification
7. Qt for Python [Электронный ресурс] : база данных. – Режим доступа : https://doc.qt.io/qtforpython/
8. Full pytest documentation [Электронный ресурс] : база данных. – Режим доступа : https://docs.pytest.org/en/latest/contents.html
9. Software access point [Электронный ресурс] : база данных. – Режим доступа : https://wiki.archlinux.org/index.php/Software_access_point

Интеграция системы смарт-контрактов в платформу Hyperledger Hyperledger Indy

А. В. Обручников

Студент магистрант

Д. И. Соломатин

Старший преподаватель

Введение

Технология блокчейн ворвалась в мир IT индустрии, породив ряд научных исследований и создав множество рабочих мест для профессионалов в сфере распределенных сетей. Сегодня можно увидеть множество вариантов применения этой технологии: финансы, право владения, интернет вещей, электронное голосование и другие.

По заявлениям ряда компаний современные централизованные системы сбора, хранения и обработки чувствительной персональной информации являются фрагментированными, не безопасными и эксклюзивными [1]. Определенно в такой позиции есть смысл, достаточно вспомнить такие громкие случаи как Facebook и Cambridge Analytica, Apple и утечки iCloud, Максима Телеком и прочие.

В отличие от подобных систем блокчейн предлагает альтернативный подход к хранению данных. Например, в идентификационной платформе Hyperledger Indy инфраструктура построена таким образом, что вся чувствительная информация о пользователе хранится на устройстве самого пользователя. Пользователь самостоятельно решает раскрывать ли ему информацию и какую

именно. Для того чтобы пользователь смог подтверждать истинность информации, используется экосистема «Self-sovereign identity» и криптографические приемы, такие как доказательство с нулевым разглашением.

Hyperledger Indy уже обладает достаточно гибкими инструментами для управления персональными данными, однако в ней отсутствует довольно популярный в других блокчейн-системах механизм смарт-контрактов. Этот механизм позволяет динамически добавлять логику в блокчейн систему, не прерывая её работы. Целостность логики зафиксирована на уровне блокчейна, что исключает возможность ее изменения или подмены.

Таким образом, целью данной работы ставится создание плагина для Hyperledger Indy с реализацией функционала смарт-контрактов.

1. Блокчейн и смарт-контракты

Стоит начать с разбора понятия блокчейн системы. По сущности, такие системы представляют собой набор нескольких серверов, общающихся между собой и клиентом с помощью специальных протоколов. В принятой терминологии сервера называются узлами, а протоколы – протоколами консенсуса. Каждый из узлов содержит базу данных, синхронизированную с остальными узлами с помощью упомянутых протоколов, которая называется блокчейн.

Отличием блокчейн систем от обычных распределенных баз данных является то, что в протоколе предусмотрена защита от некорректного поведения части узлов. То есть, если один или несколько узлов пытаются изменить данные блокчейна или эмулировать посылку некорректной транзакции от клиента, то остальные узлы должны предотвратить подобное.

В протоколах, рассматриваемых в данной работе, существует роль мастер-узла. Такой узел отвечает за то, чтобы первым принять транзакцию от клиента, обработать ее и послать другим узлам для подтверждения.

Необходимо так же описать, что из себя представляет «смарт-контракт». В 1990-х годах криптограф Ник Сабо придумал этот термин и определил его как «набор обещаний, определенных в цифровой форме, включая протоколы, в рамках которых стороны выполняют другие обещания» [2].

С тех пор концепция умных контрактов эволюционировала, особенно после введения децентрализованных блокчейнов и изобретения биткойна в 2009 году. Однако использование контрактов в экосистеме биткоина не пользовалось особой популярностью, так как язык не обладал полнотой по Тьюрингу по соображениям безопасности,

а клиентские приложения не поддерживали функционал по написанию собственных контрактов.

Более формализованный вид для смарт-контрактов определил Виталик Бутерин, создав блокчейн Ethereum, который поддерживал полный по Тьюрингу язык. Примечательно, что использование термина смарт-контракт некорректно в отношении контрактов Ethereum [3]. Они не являлись ни умными, ни юридически значимыми, однако термин закрепился.

В итоге, смарт-контракт в современном понимании можно определить следующим образом – это неизменяемая компьютерная программа, которая детерминированно выполняется в контексте виртуальной машины определенного блокчейна как часть сетевого протокола этого блокчейна.

Для выбора системы смарт-контрактов для интеграции необходимо рассмотреть существующие популярные блокчейны с такими системами, проанализировать их и выбрать наиболее подходящий вариант.

– Первым логично будет рассмотреть родоначальника смарт-контрактов Ethereum. Он является золотым стандартом в мире контрактов: огромное множество существующего кода, рекомендаций для его написания делают его почти единственным кандидатом.

– Децентрализованная платформа NEM делает ставку на масштабируемость. Для примера, Ethereum вместе со своей виртуальной машиной может достигать до 15 транзакций в секунду, в то время как у NEM по заявлениям разработчиков это число может достигать нескольких сотен в зависимости от конфигурации [4].

– Hyperledger Fabric – это блокчейн с корпоративным доступом, завязанный на модульности, что делает его крайне гибким. Отличительным плюсом Fabric является то, смарт-контракты пишутся на языке Go, являющийся компилируемым, что значительно увеличивает скорость выполнения смарт-контрактов.

Из всего вышеперечисленного лучшим выбором все равно остается виртуальная машина Ethereum. Так как помимо перечисленных достоинств, у нее имеется реализация на языке Python – py-evm [5], что значительно облегчает интеграцию в платформу Indy, написанную на этом же языке.

Таким образом, для создания плагина для Hyperledger Indy необходимо спроектировать процесс обработки смарт-контракта в рамках архитектуры и алгоритма консенсуса Indy. После того как процесс будет определен, нужно декомпозировать виртуальную машину Ethereum, а именно ее реализацию py-evm. Затем остается

импортировать полученный код виртуальной машины в код Indy в качестве подсистемы.

2. Проектирование системы смарт-контрактов в Hyperledger Indy

В Ethereum одним из основных и самым популярным языком для написания смарт-контрактов является объектно-ориентированный Solidity. Он будет использоваться в данной работе в качестве средства для описания контрактов. В этом языке контракт представлен в виде классического класса из ООП. Для контракта описывается класс, переменные которого будут сохранены в блокчейне. Функции, описанные в контракте, также сохраняются в блокчейне, но отвечают за логику, которая может оперировать содержимым хранилищ. Клиент может инициировать функции, посылая транзакции по адресу контракта, указывая сигнатуру функции и ее параметры.

Понять, как будет устроена вышеописанная логика в Indy можно на примере простого смарт-контракта, приведенного в листинге 1. С его помощью будет описано, как производится обработка транзакции и каким образом достигается согласованность выполнения смарт-контрактов в распределенной системе.

Листинг 1

Пример простейшего смарт-контракта

```
contract GetSetContract {
    string value = "Hello World!";

    function getValue() returns(string) {
        return value;
    }

    function setValue(string value) public {
        value = value;
    }
}
```

Первым этапом является инициализация контракта в сети. Клиент, имеющий достаточные полномочия, отправляет транзакцию, содержащую код контракта, в сеть. Мастер-узел принимает транзакцию, создает хранилище для переменных этого смарт-контракта и инициализирует эти переменные. В данном случае в хранилище сохранится переменная value, хранящая строку “Hello World!”. Мастер-узел хеширует созданное хранилище и отправляет хеш-сумму вместе с полученной транзакцией другим узлам.

Узлы получают сообщение от мастер-узла и независимо производят операцию, описанную в первом этапе. Далее они сравнивают полученную хеш-сумму с той, которую отправил им мастер-узел. Если

проверка прошла успешно, узлы подтверждают запись данной транзакции и рассылают подтверждение другим узлам. При наборе определенного количества подтверждений от других узлов, транзакция считается исполненной и записанной в блокчейн.

Аналогичным образом происходит и исполнение конкретных функций смарт-контракта. Например, функция «setValue» изменяет состояние хранилища. По сравнению с процессом инициализации, единственной разницей является то, что эта функция будет обращаться к уже существующему хранилищу, а не создавать новое.

В случае функции «getValue» в изменении хранилища вовсе нет необходимости. Клиенту достаточно будет получить необходимое количество подтверждений от разных узлов о текущем состоянии хранилища.

Согласно описанной архитектуры, для работы смарт-контрактов в системе Indy необходима работоспособность двух сценариев, которые приведены ниже.

Сначала стоит описать сценарий, обеспечивающий инициализацию смарт-контракта. Диаграмму, описывающую это, можно увидеть на рис. 1.

– Клиент создает транзакцию, содержащую в себе заранее подготовленный байт-код смарт-контракта. Так же он задает в ней пустое поле адреса контракта. Это сделано по аналогии с системой контрактов Ethereum, в которой транзакция по созданию контракта отправляется на специальный нулевой адрес.

– Далее созданная транзакция должна быть отправлена в блокчейн сеть. Логика обработки этой транзакции должна проверить целевой адрес контракта. Если адрес пустой, мы должны сгенерировать адрес для нового контракта. Для этого надо запросить специальное число nonce. Nonce это специальный механизм также взятый из Ethereum. В этом механизме адрес нового контракта генерируется на основании адреса отправителя, сложенного с числом nonce и взятой от них хэш суммы. Nonce представляет собой целое положительное число, изначально равное нулю. Оно увеличивается на единицу каждый раз, когда участник сети посылает транзакцию создания смарт-контракта.

– После генерации адреса контракта, необходимо сохранить переданный код контракта в блокчейн хранилище. Тем самым обеспечивается его сохранность и неизменность.

– В результате полученный адрес возвращается отправителю транзакции.

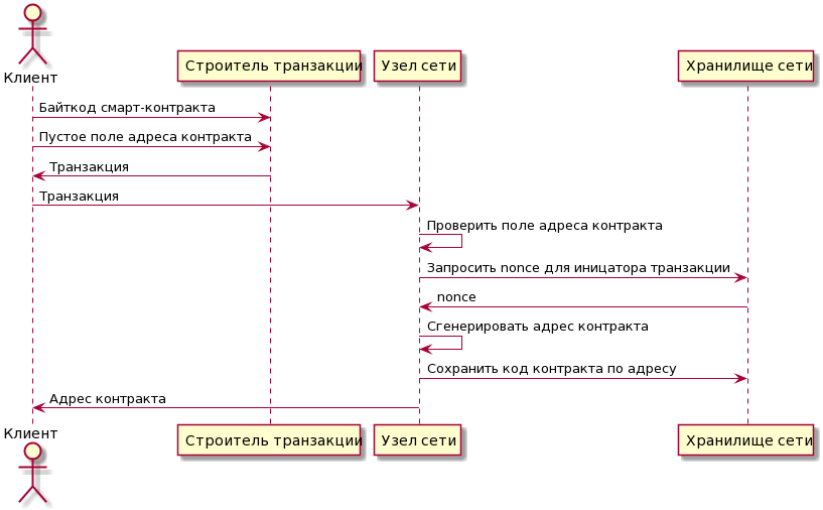


Рис. 1. Диаграмма инициализации смарт-контракта

Код, сохраняемый в блокчейн, впредь будет исполняться каждый раз, когда будет отправлена транзакция на сгенерированный адрес. Вместе с такой транзакцией отправляется специальный аргумент, состоящих из двух частей – сигнатуры функции и параметров для этой функции. Байт-код будет выполняться таким образом, чтобы переданная сигнатура вызывала определенную функцию контракта, а параметры передавались в эту функцию.

Сигнатура функции в Ethereum вычисляется следующим образом – первые четыре байта хэш-суммы Keccak (SHA-3) от имени функции, сложенной с типами параметров, перечисленных через запятую в круглых скобках.

Следующий сценарий – это вызов контракта. Диаграмма изображена на рис. 2.

– Создание транзакции, в которой указаны сигнатура функции и адрес смарт-контракта.

– Отправка транзакции в сеть.

– Получение кода контракта из хранилища блокчейна.

– Сбор кода, сигнатуры и аргументов. Передача их для выполнения в виртуальную машину.

– Получение результата вычислений и передача их клиенту.

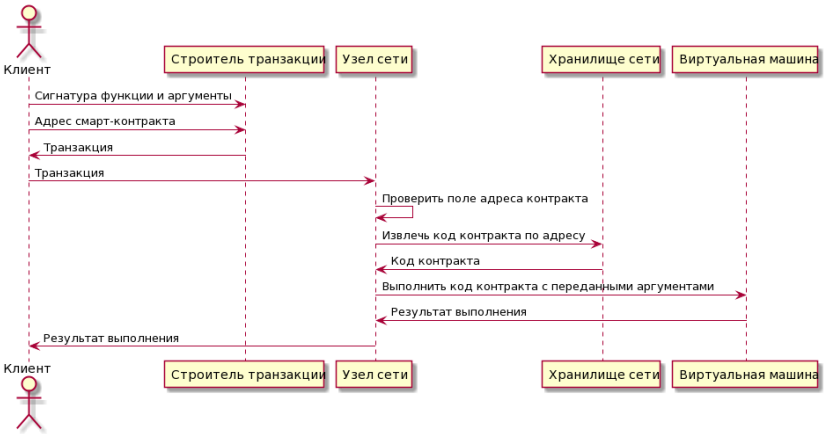


Рис. 2. Диаграмма вызова смарт-контракта

3. Интеграция смарт-контрактов в Hyperledger Indy

Основной задачей при декомпозиции py-evm было избавление от специфической логики Ethereum. В особенности, это касалось избавления от функционала взятия платы за выполнения операций внутри виртуальной машины. Так же были удалены элементы взаимодействия с блокчейном и протоколом консенсуса Ethereum. Без изменений остались вычислительные модули, системные вызовы и в целом функционал виртуальной машины, непосредственно ответственный за байт-код.

Еще одним вопросом вставшем процессе интеграции стало наличие форков Ethereum. Форк в блокчейне это версия системы с функционалом отличным от других версий. Этот термин вынесен в блокчейн системах отдельно, так как в них процесс обновления и изменения существующего функционала довольно затруднителен. В Ethereum на данный момент существует около десяти форков. В процессе интеграции система форков была отброшена, так как нововведения в основном касались элементов специфичных для Ethereum системы. Был оставлен самый первый форк «frontier», в котором и содержится весь функционал виртуальной машины.

В платформе Indy существует гибкая система плагинов, которая позволяет поддерживать новые виды транзакций без модификации ключевых частей системы. Для того чтобы добавить новую транзакцию, необходимо выполнить ряд требований:

– определить новые типы транзакций;

– определить правила проверки (валидации) новых сообщений или изменить существующие правила валидации;
– добавить логику новых транзакций, выполнить проверку логики, внесение изменений в базу данных Indy;
– выбрать хранилище для новых транзакций в существующей базе или завести новое;
– внести изменения в одно из существующих хранилищ-состояний или создать новое;
– определить новые константы;
– определить новые переменные конфигурации.

После выполнения этих требований была получена работающая подсистема смарт-контрактов, способная принимать, хранить и исполнять пользовательскую логику.

Для полученного решения можно использовать один из существующих Solidity компиляторов для создания байт-кода, например solc-js. В листинге 2 можно увидеть байт-код, скомпилированный из контракта, приведенного в листинге 1.

Листинг 2

Пример простого смарт-контракта

```
CALLDATALOADFUNCTION,                JUMPI,
PUSH1,                               RETURN,
0x0,                                 JUMPDEST,
EQ,                                  PUSH32,
PUSH1,          *'Hello World'.encode().rjust(32, b'\x00'),
0x1C,                                PUSH1,
JUMPI,                               0x0,
CALLDATALOADFUNCTION,                SSTORE,
PUSH4,                               RETURN,
*'2096'.encode(),                    JUMPDEST,
EQ,                                  PUSH1,
PUSH1,                               0x0,
0x42,                                SLOAD,
JUMPI,                               PUSH1,
CALLDATALOADFUNCTION,                0x0,
PUSH4,                               MSTORE,
*'93a0'.encode(),                    PUSH1,
EQ,                                  0x20,
PUSH1,                               PUSH1,
0x41,                                0x0,
                                     RETURN
```

Ранее, для внедрения новой функциональности в Indy, было необходимо обращаться к исходному коду системы, менять ее настройки, создавать новые хранилища и планировать сложное обновление децентрализованной системы. Система смарт-контрактов,

будучи внедренной в Indy, позволяет реализовать новую логику в системе всего лишь с помощью одного описанного контракта. Причем это возможно сделать динамически, без перезагрузки системы.

В качестве демонстрации преимуществ интегрированной системы, можно рассмотреть контракт, приведенный в листинге 3. На этом примере представлен самый простой вариант криптовалюты.

В этом контракте объявлена одна переменная, являющаяся ассоциативным массивом. Она содержит в себе пары вида «(ключ, значение)», где ключом выступает адрес аутентифицированного клиента в системе Indy, а значением количество цифровой валюты, которое есть на счету этого клиента. В контракте присутствуют две функции: giveBalance и transfer. Первая отвечает за единоразовую первоначальную выдачу средств новоприбывшему участнику, а вторая за перевод средств от одного участника к другому.

Примечательным является то, что в контракте используется глобальная переменная sender. Она представляет собой аккаунт клиента Indy, отправившего транзакцию, и дает доступ к некоторой информации о нем. Например его did – идентификатору аккаунта в системе Indy. В рамках контракта нет необходимости беспокоиться о безопасности и проверке идентификатора на подлинность. Это уже сделала система верификации Indy, проверив подпись отосланной клиентом транзакции.

Листинг 3

Пример контракта простой криптовалюты

```
contract ValueStorageContract {
    mapping (byte => uint256) balances;

    function giveBalance () returns(bool) {
        if (sender.did not in balances) {
            balances[sender.did] = 500
            return true
        }
        return false
    }

    function transfer (int did, int amount) returns(bool) {
        if (balances[sender.did] >= amount) {
            balances[sender.did] -= amount
            balances[did] += amount
            return true
        }
        return false
    }
}
```

Заключение

Результатом работы стал работоспособный прототип, рассматриваемый в качестве кандидата для внедрения в проект Indy. В данный момент он находится в процессе тестирования и анализа на предмет безопасности.

Так же, в ходе работы было проведено исследование существующих систем смарт-контрактов и виртуальных машин, на которых такие системы основаны. Была декомпозированная виртуальная машина Ethereum и интегрирована в существующую идентификационную блокчейн платформу Hyperledger Indy.

Список литературы

1. Pierre-Louis Aublin, RBFT: Redundant Byzantine Fault Tolerance [Электронный ресурс]. – Режим доступа : https://pakupaku.me/plaublin/rbft/5000a297.pdf
2. Nick Szabo, Smart Contracts: Building Blocks for Digital Markets [Электронный ресурс]. – Режим доступа : http://www.truevaluemetrics.org/DBpdfs/BlockChain/Nick-Szabo-Smart-Contracts-Building-Blocks-for-Digital-Markets-1996-14591.pdf
3. Andreas M. Antonopoulos, Mastering Ethereum [Электронный ресурс]. – Режим доступа : https://github.com/ethereumbook/ethereumbook
4. Myth or Fact? 4,000 transactions per second on the private Catapult blockchain [Электронный ресурс]. – Режим доступа : https://nemflash.io/facts-check-4000-transactions-private-catapult-blockchain/
5. A Python implementation of the Ethereum Virtual Machine [Электронный ресурс]. – Режим доступа : https://github.com/ethereum/py-evm

Алгоритмы оценки позы человека

Д. С. Палухин

Студент магистрант

М. А. Дрюченко

Доцент

Введение

В последнее время все чаще появляется необходимость отслеживания действий люди и животных в автоматическом режиме, то есть без помощи человека. Беспилотные автомобили проверяют, не собирается ли человек перейти дорогу, биологи сравнивают животных и их способности по перемещению их конечностей в пространстве, а игровые приставки считывают положение игрока. Все это происходит с помощью алгоритмов pose estimation.

Задача pose estimation заключается в точном определении ключевых точек человека в пространстве. Разнообразие сфер, где это используется, является показателем актуальности и сложности проблемы. Предполагаемое местоположение ключевой точки должно максимально точно совпадать с реальным суставом, чтобы результат мог быть правильно интерпретирован.

Формально задачу можно описать так: на вход поступает изображение x, на выходе мы должны получить вектор позы $y = (\mathsf{K}, y_i, \mathsf{K})$, $i \in \{1, \mathsf{K}, k\}$ где y_i содержит координаты i-го сустава. Таким образом мы можем представить скелет в виде графа из k вершин.

Основным подходом в последнее десятилетие являются сверточные нейронные сети. Недостаток, которым они обладают – это необходимость в однородности входных данных. Исследуемый объект должен быть по центру и того же масштаба, что и остальные объекты в выборке. Эта проблема была решена при помощи другого средства, а именно детекции человека на изображении. Обрезая исходное изображения по координатам, полученных из детектора, а после изменение размера под необходимое нам для входа, мы как раз получаем оцентрованное изображение, в котором человек занимает все основное пространство.

В данной статье будут рассматриваться алгоритмы определения ключевых точек человека в двумерном пространстве с использованием

именно нейронных сетей: Stacked Hourglass Network и Chained Predictions.

1. Stacked Hourglass Network

Данный алгоритм состоит из нескольких модулей, каждый из которых является «песочными часами». Схематично он изображен на рис. 1. Сначала изображение проходит несколько слоев сверток, после чего, выходной объект сжимается по ширине и высоте, при этом увеличиваясь по количеству каналов, а позже расширяется с уменьшением каналов. Помимо этого, сквозь каждый такой модуль прокидывается значение с выхода предыдущего модуля, что позволяет сохранять результаты с прошлых операций. Эти выходы конкатенируются и подаются на вход следующему модулю. Это позволяет учитывать зависимости изображения на любом масштабе, например, зависимость местоположение пальца относительно руки и местоположение головы относительно тела.

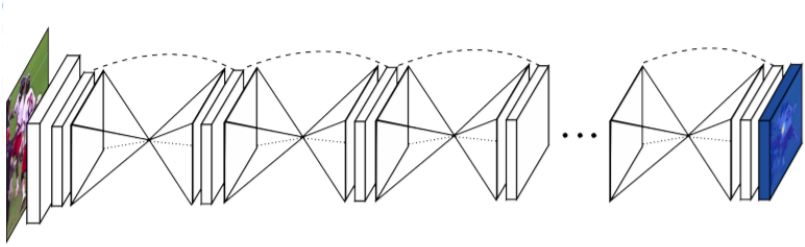


Рис. 1. Архитектура Stacked Hourglass Network

Каждый модуль данной нейронной сети представляет собой Unet-подобную архитектуру [1], продемонстрированную на рис. 2.

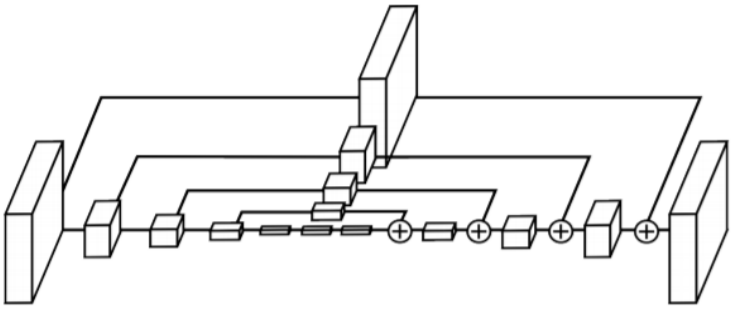


Рис. 2. Hourglass модуль

Внутри песочных часов используются только фильтры размером 3x3, что дает наилучший итоговый результат. Работа с входным изображением 256x256 требует значительного объема памяти графического процессора, поэтому самое высокое разрешение песочных часов (и, следовательно, окончательное выходное разрешение) составляет 64x64. Полная сеть начинается со сверток 7x7 с шагом 2, за которыми следует функция активации ReLU и MaxPool, чтобы снизить разрешение с 256 до 64.

Благодаря идущим подряд одинаковым модулям песочных часов сеть обрабатывает результаты выходов из каждого модуля для дальнейшей оценки и переоценки пространственных отношений более высокого порядка. В окончательной сети используются восемь модулей песочных часов.

2. Chain model

Вторая рассматриваемая модель – модель типа sequence-to-sequence [2]. Предсказания каждого выхода будут происходить последовательно, как это показано на рис. 3. Например, выход, означающий местоположения правой кисти, будет учитывать положение правого локтя.

Общая формула распределения выходов:

$$P(Y = y \mid X) = P(Y_0 = y_0 \mid X, y_0, \ldots, y_{t-1}). \quad (1)$$

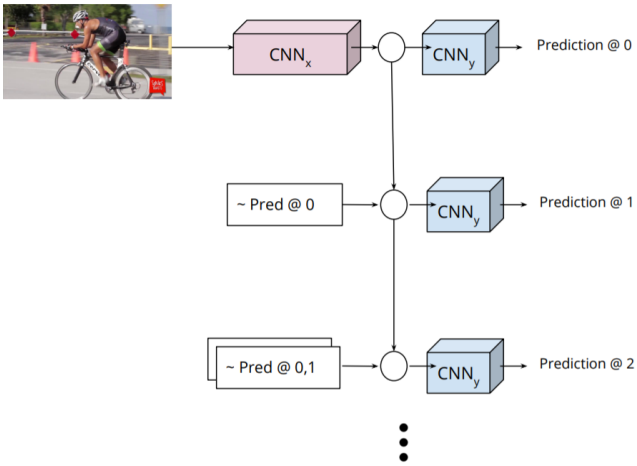


Рис. 3. Демонстрация зависимости различных выходов от предыдущих значений

В сети вход X является изображением, в то время как *t*-тая переменная Y_t – это местоположение *t*-го объекта на изображении X. Вероятность каждого шага в разложении уравнения (1) определяется через скрытое состояние h_t на шаге t, которое несет информацию о входных данных, а также о состояниях на предыдущих шагах. Кроме того, *h* включает значения $y < t$ из предыдущих шагов. Окончательная вероятность для переменной Yt вычисляется из скрытого состояния:

$$h_t = \sigma(w_t^h * h_{t-1} + \sum_{y=0}^{t-1} w_{i,t}^{t-1} * e(y_i)) . \quad (2)$$

$$P(Y_t = y_t \mid X, y_0, \mathsf{K}\ , y_{t-1}) = \text{Softmax}(m_t(h_t)) . \quad (3)$$

В приведенном выше уравнении предыдущие переменные сначала преобразуются через полную нейронную сеть e(•). Затем параметры w_t^h и $w_{i,t}^y$ линейно преобразуют предыдущее скрытое состояние и прошлые выходы, и ко всему этому применяется нелинейность σ – ReLU. Введение переменной y_{t-1} в этом уравнении гарантирует всю историю соединений. При вычислении условной вероятности y_t из h_t используется другая нейронная сеть m_t, которая дает оценки для потенциального местоположения объекта. Применяемая к этим оценкам функция Softmax, конвертирует их в распределение вероятностей по местоположениям. Начальное состояние h_0 вычисляется исключительно на основе входа $X : h_0 = CNN(X)$.

3. Данные для обучения и тестирования

Для обучения был выбран датасет MPII Human Pose [3], включающий в себя 25 тысяч изображений, на которых размечено 40 тысяч людей. 28 тысяч из людей пошли в обучающую выборку и 12 в валидационную. В данном датасете представлены люди, занимающиеся абсолютно разной деятельностью, что позволяет максимально широко рассматривать и решать задачу.

В качестве метрики был выбран PCKh, который описывает точность локализации точки с трешхолдом в 50% от размера головы человека (см. таблицу).

Таблица

Значение метрик

Модель	PCKh
ChainedPredictions	81.8
StackedHourGlass	87.6
DeepPose	54.2

4. Результаты экспериментов

Для сравнения также была взята известная нейронная сеть DeepPose [4], которая одна из первых показала возможность предсказывать ключевые точки с помощью регрессии.

Результаты обучения сети продемонстрированы на рис. 4 и рис. 5: обе сети показывают очень схожие результаты на видимых фрагментах и немного отличаются на скрытых.

Рис. 4. Результаты Stacked Hourglass Network

Рис. 5. Результаты сети Chained Predictions

Заключение

В ходе работы были рассмотрены алгоритмы ChainedPredictions и StackedHourGlass, проведен их анализ и сравнение результатов обучения. Как видно из таблицы, обе сети отлично справляются со своей задачей, но StackedHourGlass показывает лучшие результаты. Данные алгоритмы могут использоваться для определения ключевых точек человека, а впоследствии для определения действия игрока по видео.

Список литературы

1. Ronneberger, O. U-Net: Convolutional Networks for Biomedical Image Segmentation / O. Ronneberger, P. Fischer, T. Brox [Электронный ресурс]. – Режим доступа : arXiv: 1505.04597v1

2. Sutskever, I. Sequence to sequence learning with neural networks / I. Sutskever, O. Vinyals, Q.V. Le. – [Электронный ресурс]. – Режим доступа : https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

3. Andriluka, M. 2d human pose estimation: New benchmark and state of the art analysis / M. Andriluka, L. Pishchulin, P. Gehler, B. Schiele // Computer Vision and Pattern Recognition (CVPR). – 2014. – P. 3686-3693.

4. Toshev, A. Deeppose: Human pose estimation via deep neural networks / A. Toshev, C. Szegedy // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2014. – P. 1653-1660.

Интегрирование одной разложимой 7-мерной алгебры Ли

А. А. Панькова

Студент магистрант

Д. Д. Вычиков

Студент магистрант

А. В. Лобода

Профессор

Постановка задачи

Целью работы является изучение интегральных поверхностей одной вещественной алгебры Ли в четырёхмерном комплексном пространстве.

Напомним, что алгебра Ли — это линейное пространство с дополнительной билинейной антисимметричной операцией коммутирования, удовлетворяющей тождеству Якоби [1].

В работе изучается 7-мерная алгебра Ли, разложенная в прямую сумму $g_7 = g_5 + g_2$ пятимерной алгебры Гейзенберга (с двумя коммутационными соотношениями $[e_2, e_4] = e_1$ и $[e_3, e_5] = e_1$) и нетривиальной двумерной алгебры Ли (с единственным соотношением $[e_6, e_7] = e_6$). Рассматриваются алгебры векторных полей в 4-мерном комплексном пространстве, имеющие именно описанную структуру g_7. При этом каждое отдельное базисное поле в таких алгебрах представляется в виде

$$e_k = a_k \cdot \frac{\partial}{\partial z_1} + b_k \cdot \frac{\partial}{\partial z_2} + c_k \cdot \frac{\partial}{\partial z_3} + d_k \cdot \frac{\partial}{\partial z_4},\ k = \overline{1,7}, \qquad (1)$$

где a_k, b_k, c_k, d_k – аналитические функции четырех комплексных переменных z_1, z_2, z_3, z_4.

Вещественная 7-мерная гиперповерхность M называется интегральной для некоторой алгебры Ли g векторных полей, если все эти поля касаются M, т. е. для каждого базисного поля e_k этой алгебры выполняется тождество (см. [2])

$$\mathrm{Re}\,(e_k(\Phi)\big|_M \equiv 0), k = \overline{1,7}\,. \qquad (2)$$

Здесь Φ – определяющая функция поверхности M, т. е.

$$\{M = \Phi(z_1, z_2, z_3, z_4) = 0\}. \qquad (3)$$

Задачей настоящей работы является получение невырожденных (существенно зависящих от всех четырех комплексных переменных) орбит у обсуждаемой алгебры.

1. Реализация абстрактных алгебр в виде алгебр векторных полей

Любое из базисных полей запишем следующим образом: $e_k = (a_k(z), b_k(z), c_k(z), d_k(z))$, где $z = (z_1, z_2, z_3, z_4)$.

Для удобства работы с обсуждаемой алгеброй предлагается упростить базис ее реализации в $\mathbb{J}^4$. Для такого упрощения можно использовать наличие в алгебре $g_7 = g_5 + g_2$ четырехмерной абелевой подалгебры $h = < e_1, e_2, e_3, e_6 >$. Имеет место утверждение (см. [3]):

Лемма. Если 7-мерная алгебра векторных полей в $\mathbb{J}^4$ имеет невырожденные орбиты и 4-мерную абелеву подалгебру h, то

фиксированный базис e_1, e_2, e_3, e_6 такой подалгебры можно считать имеющим один из следующих видов:

I тип базиса: $e_1 : (1,0,0,0)$, $e_2 : (0,1,0,0)$, $e_3 : (0,0,1,0)$, $e_6 : (0,0,0,1)$;

II тип базиса: $e_1 : (1,0,0,0)$, $e_2 : (0,1,0,0)$, $e_3 : (a_3(z_3), b_3(z_3), 0, d_3(z_3))$, $e_6 : (0,0,0,1)$;

III тип базиса: $e_1 : (1,0,0,0)$, $e_2 : (a_2(z_3), 0, 0, d_2(z_3))$, $e_3 : (0,1,0,0)$, $e_6 : (0,0,0,1)$.

Для нахождения возможных интегральных гиперповерхностей мы обсудим все три типа базисов 4-мерной подалгебры. Для каждого из этих типов мы сначала упростим остальные базисные поля нашей 7-мерной алгебры.

2. Упрощение базисов обсуждаемых алгебр

В первую очередь рассчитаем коммутаторы между всеми полями из абелевой подалгебры и теми полями, которые в подалгебру не вошли.

Начнем с первого типа базиса. Здесь имеем коммутатор

$$[e_1, e_4] = 1 \cdot \left(\frac{\partial a_4(z)}{\partial z_1}, \frac{\partial b_4(z)}{\partial z_1}, \frac{\partial c_4(z)}{\partial z_1}, \frac{\partial d_4(z)}{\partial z_1} \right) = 0 . \qquad (4)$$

На этом основании можно сказать, что $a_4(z), b_4(z), c_4(z), d_4(z)$ не зависят от z_1. Аналогичным образом посчитаем остальные коммутаторы и получим, что:

e_5, e_7 не зависят от z_1 и z_2; e_4, e_7 не зависят от z_3; e_4, e_5 не зависят от z_4;

$[e_2, e_4] = e_1$, тогда $b_4(z), c_4(z), d_4(z)$ не зависят от z_2, а $\frac{\partial a_4(z)}{\partial z_2} = 1$;

$[e_2, e_4] = e_1$, тогда $b_5(z), c_5(z), d_5(z)$ не зависят от z_3, а $\frac{\partial a_5(z)}{\partial z_3} = 1$;

$[e_2, e_4] = e_1$, тогда $a_7(z), b_7(z), c_7(z)$ не зависят от z_4, а $\frac{\partial d_7(z)}{\partial z_4} = 1$.

Опираясь на полученные результаты, можно сказать, как выглядят поля: e_4, e_5, e_7: $e_4 = (A_4 + z_2, B_4, C_4, D_4)$, $e_5 = (A_5 + z_3, B_5, C_5, D_5)$, $e_7 = (A_7, B_7, C_7, D_7 + z_4)$.

В этой записи заглавными латинскими буквами обозначены комплексные константы. Далее рассчитаем попарные коммутаторы полей, не входящих в абелеву подалгебру, а именно: $[e_4, e_5]$, $[e_4, e_7]$ и $[e_5, e_7]$. Распишем один из коммутаторов:

$$[e_4, e_5] = C_4(1,0,0,0) - B_5(1,0,0,0) = 0 \Rightarrow C_4 = B_5 . \quad (5)$$

Вычисляя аналогичным образом остальные коммутаторы, получим:

$$[e_4, e_7] = D_4(0,0,0,1) - B_7(1,0,0,0) = 0 \Rightarrow D_4 = B_7 = 0 ; \quad (6)$$

$$[e_5, e_7] = D_5(0,0,0,1) - C_7(1,0,0,0) = 0 \Rightarrow D_5 = C_7 = 0 . \quad (7)$$

Применив аффинные преобразования вида

$$z_k^* = z_k + Const, \quad (8)$$

получим базисные векторные поля алгебры g_7 в следующем виде

$e_1 : (1,0,0,0); e_2 : (0,1,0,0) ; e_3 : (0,0,1,0) ; e_6 : (0,0,0,1); e_4 : (z_2, B_4, C_4, 0);$

$e_5 : (z_3, C_4, C_5, 0); e_7 : (A_7, 0, 0, z_4).$

В данном варианте набора векторных полей, заданных в явном виде, противоречий не обнаружено, так что исследуемую алгебру Ли возможно проинтегрировать.

3. Интегрирование алгебр первого типа

Дифференцируем определяющую функцию Φ первыми тремя векторными полями. В результате для каждого $i = \overline{1,4}$ получаем:

$$\frac{1}{2} \cdot \mathrm{Re}\left\{ \frac{\partial \Phi}{\partial x_i} - i \cdot \frac{\partial \Phi}{\partial y_i} \right\} = 0 \Rightarrow \frac{1}{2} \cdot \frac{\partial \Phi}{\partial x_i} = 0 \Rightarrow \frac{\partial \Phi}{\partial x_i} = 0 . \quad (9)$$

Данное равенство позволяет сделать вывод о том, что Φ не зависит от x_i. Если при этом Φ также не зависит от y_4, то мы имеем вырожденный случай. Значит, для искомых невырожденных поверхностей Φ зависит от y_4, тогда можно считать, что $\frac{\partial \Phi}{\partial y_4} \neq 0$. С учётом этого представим функцию Φ в следующем виде:

$$\Phi = -y_4 + F(y_1, y_2, y_3) . \quad (10)$$

Применив формулу (9), а также равенство $y_4 = F$ к векторным полям e_4, e_5, e_7, выведем следующую систему уравнений

$$\begin{cases} y_2 \dfrac{\partial F}{\partial y_1} + B_{42} \dfrac{\partial F}{\partial y_2} + C_{42} \dfrac{\partial F}{\partial y_3} = 0 \\ y_3 \dfrac{\partial F}{\partial y_1} + C_{42} \dfrac{\partial F}{\partial y_2} + C_{52} \dfrac{\partial F}{\partial y_3} = 0 \\ A_{72} \dfrac{\partial F}{\partial y_1} - F = 0. \end{cases} \quad (11)$$

Решив систему (11), получим следующий вид функции при ненулевом знаменателе:

$$F = C \cdot e^{\frac{(C_{42}^2 - B_{42}C_{52})(\frac{y_2^2}{2} - B_{42}y_1) + \frac{(C_{42}y_2 - B_{42}y_3)^2}{2}}{A_{72}(C_{42}^2 - B_{42}C_{52})}}. \quad (12)$$

После переобозначений констант и линейных замен переменных уравнение (12) приводит к следующему виду искомых поверхностей:

$$\Phi = -y_4 + \ln y_1 \pm y_2^2 \pm y_3^2. \quad (13)$$

4. Рассмотрение базиса второго типа

Исследуя результаты коммутирования, как и в предыдущем случае, на первом этапе рассмотрения базиса придем к виду векторных полей: $e_4 : (a_4(z_3) + z_2, b_4(z_3), c_4(z_3), d_4(z_3))$,

$e_5 : (a_5(z_3), b_5(z_3), c_5(z_3), d_5(z_3))$, $e_7 : (a_7(z_3), b_7(z_3), c_7(z_3), d_7(z_3) + z_4)$.

Обратим внимание на векторное поле e_5. Его третья координата определенно не может быть тождественно нулевой, поскольку в алгебре g_5 присутствует равенство $[e_3, e_5] = e_1$. Выпрямим это поле по третьей координате [3] и получим $e_5 = (0,0,1,0)$. Так как в алгебре g_5 возможна перестановка базисных полей e_3 и e_5, то второй тип базиса превращается в первый. Проводить дальнейшие поиски интегральных поверхностей для второго типа нет необходимости.

5. Вырождение поверхностей для базиса третьего типа

Рассмотрим третий и последний тип базиса: рассуждая уже обозначенным способом, увидим, что условно-неизвестные поля в случае базиса 4-мерной подалгебры третьего типа будут такими: $e_4 : (a_4(z_3), b_4(z_3), c_4(z_3), d_4(z_3))$, $e_5 : (a_5(z_3) + z_2, b_5(z_3), c_5(z_3), d_5(z_3))$, $e_7 : (a_7(z_3), b_7(z_3), c_7(z_3), d_7(z_3) + z_4)$.

В этом случае рассмотрим поле e_4 и равенство $[e_2, e_4] = e_1$, запрещающее нуль в третьей координате e_4.

После упрощений, аналогичных описанным выше, получаем следующий вид базиса всей 7-мерной алгебры: $e_1 : (1,0,0,0)$, $e_2 : (-z_3, 0, 0, D_2)$, $e_3 : (0,1,0,0)$, $e_6 : (0,0,0,1)$, $e_4 : (0,0,1,0)$, $e_5 : (z_2, B_5, C_5, D_5)$, $e_7 : (A_7, B_7, C_7, +z_4)$.

Из равенства $[e_2, e_5] = 0$ легко следует, что $C_5 = 0$, а из равенства $[e_2, e_7] = 0$ аналогично получаем $C_7 = 0$.

Замечание. При наличии шести нулей в одной из координат (например, в третьей) набора семи базисных полей алгебры, любая

интегральная поверхность такой алгебры не зависит от всех остальных координат, а значит, является вырожденной.

С учетом этого замечания для третьего типа базиса подалгебры h все интегральные поверхности исходной алгебры g_7 оказываются вырожденными.

Заключение

В четырёхмерном комплексном пространстве получено полное описание невырожденных интегральных поверхностей одной разложимой вещественной 7-мерной алгебры Ли. В отличие от большого семейства разложимых алгебр вида $g_7 = g_5 + g_2$, не имеющих невырожденных орбит (см. [4], а также [5]), прямая сумма 5-мерной алгебры Гейзенберга и двумерной нетривиальной алгебры обладает такими орбитами.

Список литературы

1. Серр, Ж.-П. Алгебры Ли и группы Ли / Ж.-П. Серр. – М. : Мир, 1969. – 376 с.
2. Лобода, А. В. Однородные строго псевдо-выпуклые гиперповерхности в $\mathbb{J}^3$ с двумерными группами изотропии / А. В. Лобода // Математический сборник. – 2001. – № 12. – С. 3-24.
3. Loboda, A. V. On the orbits of nilpotent 7-dimensional Lie algebras in 4-dimensional complex space / A. V. Loboda, R. S. Akopyan, V. V. Krutskikh // Журнал СФУ. Математика и физика. – Красноярск, 2020. –Т. 13. – №3. – С. 360-372.
4. Хацкевич, Д. И. О вырождениях орбит разложимых 7-мерных алгебр Ли / Д. И. Хацкевич, Г. В. Зинькевич, А. В. Лобода // Сборник студенческих научных работ ФКН ВГУ. – Воронеж, 2020. – Вып. 14. – С. 290-295.
5. Атанов, А. В. Разложимые пятимерные алгебры Ли в задаче о голоморфной однородности в $\mathbb{J}^3$ / А. В. Атанов, А. В. Лобода // Итоги науки и техники. Современная математика и её приложения. Тематические обзоры. – 2019. – Т. 173. – С. 86-115.

Преимущества применения шаблонов языка C++ для реализации интроспекции

Л. А. Прохорченко

Студент

Д. И. Соломатин

Старший преподаватель

Введение

Если вы участвуете в разработке масштабного приложения, в котором есть множество разнотипных данных, то рано или поздно вы столкнётесь с необходимостью выполнять какие-либо обобщённые операции над этими типами данных. Если это веб-приложение, то вам может понадобиться прочитать множество разных сущностей из базы данных. Если вы разрабатываете standalone приложение, может возникнуть необходимость сериализовать результат работы пользователя, например какой-нибудь проект с множеством данных.

Для выполнения обобщённых действий в действующих языках программирования придумано множество различных механизмов, подходящих для различных ситуаций. Однако, многие простые «классические» методы, например, использование полиморфизма для классов-данных, хороши в теории, но не в общем случае разработки приложений. Чтобы найти оптимальный в наибольшем числе случаев метод реализации обобщённых действий с данными, нужно выбрать наиболее абстрактный и вариативный круг задач, для которых они применяются. Положим, что в общем случае приложение уже находится не в начале этапа разработки, а потому может быть масштабным, а также может содержать множество модулей, которые невозможно изменить или даже посмотреть исходники.

Задачей предварительного исследования являлось обоснование использования механизма интроспекции для решения подобных задач. В данной работе рассматриваются способы применения интроспекции в языке программирования C++, подходящие для максимально широкого класса приложений.

1. Удобства применения интроспекции

Перед обоснованием применения соответствующего механизма, сначала следует обратиться к его определению. Интроспекция – один из

видов самомодифицирующегося кода (далее *СМК*). СМК – определённый приём программирования, позволяющий программе сгенерировать дополнительный или изменить существующий код программы при её запуске или во время исполнения. СМК является частью метапрограммирования. Механизм интроспекции, в частности, позволяет получать доступ к полям и методам классов во время выполнения программы. Это позволяет получить из кода доступ к некоторой модели каждой введённой структуре данных (здесь и далее *метамодели*).

Во введении используется понятие обобщённых операций и, для ясности, его суть тоже нужно раскрыть. Здесь и далее под обобщёнными операциями будут подразумеваться действия с множеством сущностей различных классов. Эти действия должны по-своему выполняться для каждого из типов данных.

Во многих ситуациях, когда возникает такая программная задача, достаточно реализации полиморфизма, необходимая операция становится методом каждого из этих классов, а затем по-своему определяется для каждого конкретного класса. В большинстве строго-типизированных языков программирования, это обозначает также необходимость ввести для всех этих классов общий класс-предок.

Однако когда речь идёт об операциях над классами *данных*, тем более в комплексном приложении, всё немного иначе. Следует начать с того, что согласно грамотности архитектуры проектирования межклассовых взаимодействий, классы-данные должны стоять особняком, и не знать об операциях, которые над ними будут выполняться. Это порождает необходимость создавать целые иерархии обработчиков для каждого обобщённого действия. Например, абстрактный сериализатор и его подвид для каждого из классов-данных, который необходимо сериализовать. Их число растёт с числом появляющихся в проекте типов, для них необходимо писать тесты, а при изменении самого класса данных все соответствующие обработчики должны подвергаться сложным изменениям. При недостаточной документации или малой связности разработчиков в команде, весь такой программный модуль становится обильным источником ошибок, багов и нестыковок.

А вот механизм интроспекции может показать себя отлично в задаче проведения обобщённых операций над классами данных. Ведь, в отличие от общих методов в прочих классах, действия с данными всегда сводятся к работе с полями. Если записывать экземпляр класса в формат JSON, например, то так или иначе потребуется пройти по всем его полям и конвертировать их, причём для однотипных полей будет вызван

один и тот же вид конвертера. А если для каких-то полей класса данных нужно создать форму редактирования, то для каждого из них нужно создать определённый элемент управления. И чаще всего для всех текстовых полей класса это будет форма, позволяющая ввести текст.

То есть, обобщённые операции над данными в результате всегда сводятся к применению операции к каждому конкретному полю, и зависят более от типа значения этого поля, чем от того, какому классу оно принадлежит. И представить каждый класс одинаковым интерфейсом, через который можно взаимодействовать с каждым из полей и его типом, можно как раз при помощи интроспекции. Таким образом, обобщённые операции будут по-своему определяться для типов данных, а не для каждого из классов содержащих их структур. Например, не будет необходимости писать отдельную реализацию для записи в файл каждого из видов сущности работника предприятия. Достаточно будет определить способы записать в файл данные типа строка и типа число. Алгоритм для записи каждого типа сотрудника будет также обобщённым. Все их текстовые поля будут записаны одним образом, а все числовые – другим. Пропадёт необходимость писать отдельные классы-обработчики, составлять для них отдельные тесты, а также в случае введения нового типа работника или изменении описания уже существующего больше не нужно будет менять множество существующих сопряженных классов. Места, в котором потенциально можно совершить множество трудно исправимых ошибок, больше не будет.

2. Интроспекция

Применение механизма интроспекции для вышеописанных действий с данными для общего вида приложений имеет свои преимущества, в сравнении с прочими. По этой причине во многих актуальных языках программирования уже есть встроенная интроспекция. В первую очередь, это естественно для языков, работающих по принципу виртуальных машин, это, например, ECMAScript и другие динамически типизированные языки.

Среди строго типизированных языков примеров тоже не мало. В высоко абстрактных языков C# и Java есть рефлексия [1] и вспомогательные конструкции Class, Method, Field, соответственно. А первые попытки реализации можно найти ещё в языке Forth.

Однако в C++ встроенной интроспекции нет. Поэтому в данной работе проводится обзор и сравнения различных методов её реализации. Есть три наиболее действенных и простых метода создания метамоделей классов данных:

1. объектная модель;

2. кодогенерация;
3. обобщённое программирование.

Рассмотрим их преимущества и недостатки в рамках C++, применяя их реализацию и использование к определённому ранее спектру абстрактных приложений.

Под объектной моделью подразумевается архитектура данных, центральной осью которой является особый класс-объект. Такую структуру легко найти в высоко абстрактных языках программирования, например в C#. Также, их можно встретить в некоторых фреймворках для C++, например Qt Framework [2]. Программная реализация подразумевает существования общего предка всех классов, для которых программист хочет предусмотреть интерфейс доступа к полям. В этом классе-объекте хранится хэш-множество полей и значений (например, хэш-множество из QVariant для QObject). Достоинством такого подхода является простая реализация и возможность задать любой внешний интерфейс для доступа к полям и даже методам. При должной повсеместности он применим к большему множеству классов, чем просто структуры данных, например, можно систематизировать и классы-обработчики, формы, сборки, версии и многое другое. Однако, для полноценного применения этого метода необходимо соблюдение его фундаментальности. На этом принципе должна быть построена вся система, то есть, соответствующее решение должно быть принято уже на этапе проектирования приложения. А значит, в общем случае, для начатого приложения этот подход может не подойти, так как не всегда есть возможность переписать всё приложение.

Под кодогенерацией в данной работе понимается механизм, позволяющий использовать некоторые пометки в коде, которые в момент компиляции будут трансформированы в необходимый соответствующий код. Чаще всего, реализации такого подхода вырастают в полноценные расширения языка. В случае C++ это, например, расширение MetaC++ [3]. Такой подход требует расстановки специальных меток или тэгов внутри существующего кода, которые будут отправными точками для преобразования кода. В отличие от объектной модели, этот подход не требует полного переписывания архитектуры приложения, ведь его можно применять точечно. Однако, с внесением меток в классы данных, потребуется менять и прочие методы, которые с ними работают, а потому суммарное число изменений все ещё может быть велико. Этот подход также не может претендовать на универсальность. К тому же, большинство меток имеют плохо читаемый синтаксис, а использование расширения языка приводит к появлению дополнительной зависимости всего проекта.

Последняя альтернатива это обобщённое программирование. В C++ оно реализуется посредством подсистемы шаблонов. В отличие от рассмотренной кодогенерации, шаблоны – полноценная часть стандарта C++ и их использование не внедряет дополнительных зависимостей. Подсистема шаблонов определяется по Тьюрингу как полный чистый функциональный язык, но многие избегают её из-за сложности использования и отладки. Основное преимущество системы шаблонов – возможность создать метаописания классов, не изменяя их исходный код. Она работает с обобщением типов, для которых имеет значение лишь описание методов и полей, а не их реализация. С последними стандартами языка (начиная с C++11) появилась даже возможность проводить усложнённое инстанцирование шаблонов во время компиляции [4], что должно позитивно сказываться на скорости работы программы.

Далее в работе будет предложена архитектура реализации механизма метаопределений классов на шаблонах. Её использование возможно без внесения изменений в исходный код проекта, что делает данный подход применимым к наибольшему спектру различных приложений.

3. Архитектура метамоделей на шаблонах

Для реализации интроспекции необходимо обеспечить интерфейс для получения и установки значений атрибутов класса. Чаще всего эти операции выполняются через специальные методы – геттеры и сеттеры, поэтому основой метамодели будут группы указателей на эти методы.

Общие наиболее распространённые виды геттеров и сеттеров приведены в листинге 1.

Листинг 1

Обобщенный вид геттеров и сеттеров типа T для класса Class

```
template <typename Class, typename T>
struct Method {
 using Pointer = T Class::*;
 using SetterByValue = void (Class::*)(T);
 using SetterByConstRef = void (Class::*)(T const&);
 using GetterByValue = T (Class::*)() const;
 using GetterByRef= T & (Class::*)();
 using GetterByConstRef = T const& (Class::*)() const;
};
```

Имеет смысл определить поле типа T класса Class как список указателей на все ассоциируемые с ним геттеры и сеттеры. Эти данные будут статическими, так как все геттеры и сеттеры известны заранее и не могут измениться в ходе выполнения программы. В листинге 2

введена структура Member<T, Class>, обозначающее поле класса. Дополнительные используемые для неё типы – название *N* и кортеж зарегистрированных для него указателей на геттеры и сеттеры *M*. Нужно обратить внимание на то, что простой массив не подходит для хранения указателей – ведь у методов различные типы. Также, для задачи имени поля используется статическая строка, но рассмотрение её реализации выходит за рамки данной работы.

Листинг 2

Поле класса в метамодели

```
template <typename N, typename Class, typename T, typename M>
struct Member {
 using ValueType = T;
 using MethodsTupleType = M;
 static inline constexpr N name();
 static inline constexpr M methods();
 constexpr Member() {}
};
```

Единственные данные, которые хранит структура Member – статические указатели на методы. Тогда метамодель класса можно представить структурой, хранящей список полей. Структура *ClassDefinition* представленна в листинге 3. С помощью статических функций *nameOf* и *membersOf* в коде возможно «зарегистрировать» необходимые классы, тем самым создав для них метамодель.

Листинг 3

Метамодель класса

```
template <typename Class>
struct ClassDefinition {
 static inline constexpr auto name(){
 return nameOf<Class>();
 }
 static inline constexpr auto members(){
 return membersOf<Class>();
 }
};
```

Чтобы создать метамодель класса, достаточно подключить необходимый класс заголовков и переопределить обобщённый метод *membersOf*<*Class*>. Например, для класса *Home* в листинге 4, регистрация будет выглядеть так, как показано в листинге 5. Таким образом, автодедукция шаблонов в C++ найдёт реализацию метода, оставленную программистом, и использует определённые в ней указатели методов для вычислений.

Листинг 4

Класс данных типа Home

```
class Home {
public:
 Home() = default;
 string getName() const;
 string &getNameReference();
 void setName(const string &name);
 vector<int> getLiveCount() const;
 void setLiveCount(const vector<int> &count);
private:
 vector<int> _lc;
 string _n;
};
```

Листинг 5

Регистрация метамодели класса Home

```
#include “meta_define.h”;

template <> constexpr inline auto membersOf<Home>(){
 return members(
 member<C, int>(
 "name"_static,
 &Home::getName,
 &Home::getNameReference,
 &Home::setName),
 member<C, vector<int>>(
 "lc"_static,
 &Home::getLiveCount,
 &Home::setLiveCount)
}
```

Регистрацию можно объявить в любом .h файле. Для осуществления механизма интроспекции по отношению к данному классу его необходимо будет подключить. В листинге 6 обозначен способ доступа к данным экземпляра класса *Home*. В строке 1 вызывается его метамодель, и теперь можно использовать его поля, обращаясь к ним по именам. Методы *get* и *set* представляют собой поиск и вызов геттера и сеттера по значению соответственно. В строке 2 происходит копирование значения поля имени, а в строке 3 установка значение поля количества жильцов.

Этот подход не требует внесения изменений в код класса, для которого создаётся метамодель, а все обращения к полям происходят во время компиляции программы. Это обозначает, что вызов метода непосредственно выигрывает лишь на несколько системных вызовов. Основным ускорителем в данном случае выступает статический поиск

методов в кортежах методов и полей в кортежах полей. Рассмотрение их реализаций сопряжено с детальным анализом кода, а потому не будет приведено в данной работе.

Листинг 6

Использование метаопределения класса Home

```
#include "home_meta_define.h";
#include "meta_model_operations.h";

Home home;

constexpr auto metahome = instance(home); // [1]
const string name = metahome["name"_static].get(); // [2]
metahome["lc"].set({1, 2, 3}); // [3]
```

Впоследствии для любой обобщённой операции можно будет задать метод, единый по логике для типов данных и классов-данных, их содержащих. В листинге 7 представлен рекурсивный проход по всем полям для любого класса, с вызовом соответствующей функции.

Листинг 7

Рекурсивный проход по полям

```
template <typename Class, typename F>
void apply(const Class &c, F&& function){
 if constexpr (not isRegistred<Class>())
 function(c);
 else
 Operations::forAllMembers<Class>(
       [function](const auto& member){
              apply (member);
       });
}
```

Заключение

Анализ, проведённый в ходе работы, обосновывает необходимость применения интроспекции в ряде задач. Также, было проведено сравнения способов реализации этого механизма для языка C++, и была приведена архитектура модуля интроспекции с применением подсистемы шаблонов, подходящая для наиболее широкого круга производственных программных задач.

Список литературы

1. C# Reflection [Электронный ресурс] : документация. – Режим доступа : https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/reflection

2. Документация QObject фреймворка QtFramework [Электронный ресурс] : документация. – Режим доступа : https://doc.qt.io/qt-5/qobject.html#details

3. Lilis, Y. Meta C++: an extension layer for multi-stage generative metaprogramming [Electronic resource] / Yannis Lilis, Anthony Savidis // Journal of Object Technology. – 2019. – Vol. 18. – Issue 1. – P. 1-44. – Режим доступа : http://www.jot.fm/issues/issue_2019_01/article1.pdf

4. C++ constexpr explanation [Electronic resource] : article. – Access mode : https://www.modernescpp.com/index.php/constexpr-functions

Планирование проектов с ресурсными ограничениями

Н. И. Пугач

Студент магистрант

И. В. Илларионов

Доцент

Введение

Планирование множества проектов, которые выполняются одновременно в динамической среде, остается главной проблемой в управлении проектами. С одной стороны, жесткие ограничения должны быть удовлетворены выполнимыми графиками, включая предшествующие задачи и объемы ресурсов. С другой стороны, проекты зачастую выполняются в динамических средах с различными источниками неопределенности, где требуется оперативное перепланирование, которое может влиять на другие проекты из-за их взаимозависимости. Хотя классические математические подходы помогают справиться с проблемами планирования с низкой сложностью или в относительно статической среде, имеется ограничение в описании взаимоотношений в многопроектных средах и адаптации анализа к динамическим изменениям [1].

Агентно-ориентированный подход был определен как эффективный подход для проблем планирования с высокой сложностью [2, 3]. Однако, многие существующие исследования сконцентрировали внимание на алгоритмах планирования в режиме «офлайн». Очень немного исследований поддерживают онлайновое перепланирование, которое вызывается изменениями «на лету» из-за непредсказуемых помех или пользовательского влияния. Кроме того, обычно для обычных

алгоритмов планирования неэффективно управлять частым перепланированием в режиме «онлайн» из-за высокой сложности. Одним из перспективных направлений является объединение многоагентных подходов с техниками оптимизации [3], так как объединение методов оптимизации с логикой поведения агентов увеличит их интеллект, что приведет к оптимальному принятию решений как для краткосрочных, так и долгосрочных целей компании.

В этой работе предложена агентно-ориентированная система моделирования, которая должна помочь пользователям грамотно планировать несколько проектов с ресурсными ограничениями, отслеживать процесс их выполнения и обеспечить поддержку для принятия мгновенных решений.

Агентно-ориентированная система моделирования для планирования проектов с ресурсными ограничениями

Для иллюстрации предложенной системы была использована реальная программа разработки промышленного продукта. Эта программа состоит из нескольких проектов, выполняющихся одновременно, и цель – выполнить все проекты в срок. Каждый проект назначается определенному разработчику, который будет реагировать на сбои, возникающие при выполнении. Проект состоит из нескольких задач, требующих особого подхода. В свою очередь каждая задача содержит от десятков до сотен подзадач. Во время выполнения задачи все подзадачи выполняются с использованием определенных инструментов параллельно каждый день. Инструменты могут стать неисправными во время выполнения и привести к ошибкам в подзадачах, когда число неисправных инструментов достигнет предела. Любой сбой подзадачи рассматривается как нарушение, которое приведет к прекращению всей программы, что потребует поддержки принятия решений.

На рисунке показана структура предлагаемой системы моделирования, в которой входные данные, поведение встроенных агентов и логика программы доступны пользователям через интерактивный графический пользовательский интерфейс. При моделировании системы используются как объекты, так и агенты. Задачи, подзадачи и средства моделируются как объекты, поскольку они не обладают автономным поведением. С другой стороны, разработчики и проекты моделируются как агенты разработчиков и проектов. Каждому агенту разработчика назначается один или несколько проектов. В их обязанности входит мониторинг хода выполнения проекта, корректировка постановки задач и координация с другими разработчиками для устранения сбоев и конфликтов ресурсов. Каждый

проектный агент отвечает за ведение своих собственных спецификаций, а также за обновление визуализации своего графика.

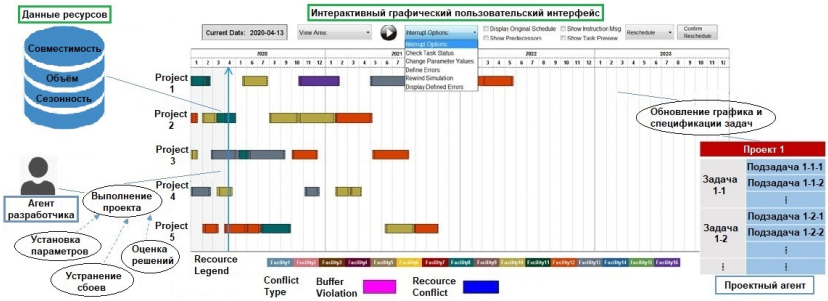


Рисунок. Структура предложенной системы моделирования

Предложенное моделирование было реализовано в AnyLogic® 7 Multi Simulation Software. Середина рисунка представляет собой скриншот запущенной симуляции в интерактивном графическом пользовательском интерфейсе. Дизайн интерфейса основан на диаграмме Ганта, чтобы свести к минимуму кривую обучения пользователей традиционных инструментов управления программами, таких как Microsoft Project. В дополнение к обеспечению доступности основной информации о проекте в обычных диаграммах Ганта, наш улучшенный дизайн также отображает взаимосвязи «задача-объект» и отклонение графика от текущего времени. Вертикальная линия синего цвета, представляющая настоящий период времени, перемещается вместе с прогрессом моделирования, чтобы дать обзор текущего состояния программы. Каждый объект обозначается определенным цветом, который соответствует цвету задач, которые его запрашивают.

Пользователям доступно несколько вариантов прерывания симуляции. Например, пользователи могут делать изменения параметров «на лету» без перезапуска симуляции. Кроме этого, система включает экспериментирование с пользователем в цикле двумя способами:

1. определение сбоев подзадач;
2. ручная настройка расписаний для будущих задач.

Если пользователи хотят оценить, как программа будет действовать, если определенные задачи не будут выполнены в определенные дни, они могут запланировать их до или во время запуска моделирования. Если кликнуть по незавершенной задаче, появится всплывающий список подзадач, чтобы пользователи могли выбрать

любые подзадачи, которые должны быть выполнены до завершения задачи. Ручное планирование достигается путем перетаскивания будущих задач. Как только пользователи закончили с ручным перепланированием, им нужно будет нажать кнопку «Confirm the Schedule», и система проверит наличие новых конфликтов ресурсов, прежде чем возобновить моделирование с новым расписанием. Конфликт ресурсов может быть решен с помощью ручного перепланирования или встроенного решателя оптимизации на основе Choco 2.1.5 [4].

В случае «онлайн» сбоя разработчики могут решить его двумя способами:

1. повторно выполнить задачу с невыполненной подзадачей;
2. переместить невыполненную подзадачу в последующие задачи.

При использовании первого способа невыполненная задача будет перезапущена через один день после даты отказа. В зависимости от дня возникновения ошибки перепланирование неудавшейся задачи может повлиять на последующие задачи в различной степени. Для второго способа разрешения сбоя разработчик должен выбрать целевую задачу из нисходящего потока для размещения неудачной подзадачи. Квалифицированная целевая задача должна иметь требуемое средство, которое обладает способностью инструмента выполнить невыполненную подзадачу. Поскольку задержка, вызванная перемещением неудачной подзадачи, очень мала по сравнению с задержкой, вызванной повторным выполнением задачи, мы предполагаем, что ее можно игнорировать в смоделированной программе. Однако это решение приведет к дополнительным затратам из-за изменения целевой задачи. Поэтому решение пользователей между двумя способами разрешения сбоя в режиме реального времени может варьироваться в зависимости от их предпочтения относительно более короткого времени завершения или меньшей стоимости.

Заключение

Данная работа посвящена разработке интерактивной системы моделирования на основе агентов для обеспечения поддержки принятия решений при планировании нескольких проектов с ресурсными ограничениями. Реализованная в программном обеспечении AnyLogic® агентно-ориентированная система моделирования предлагает интерактивный графический пользовательский интерфейс, который позволяет изменять параметры моделирования «на лету», определять сбои в конкретных задачах, корректировать расписание будущих задач. Имеется несколько направлений для будущих исследований. Например, разработка механизма согласования для автоматизации процедуры

поиска наилучшей доступной целевой задачи для перемещения невыполненной подзадачи. Кроме того, данную систему можно усложнить путем ослабления некоторых ограничений.

Список литературы

1. Arauzo, J. Simulating the dynamic scheduling of project portfolio / J. Arauzo, J. Pajares, A. Lopez-Paredes // Simulation Modelling Practice and Theory. – 2010. – Vol. 18. – № 10. – P. 1428-1441.

2. Adhau, S. A multi-agent system for distributed multi-project scheduling: An auction-based negotiation approach / S. Adhau, M. Mittal, A. Mittal // Engineering Applications of Artificial Intelligence. – 2012. – Vol. 25. – № 8. – P. 1738-1751.

3. Scheduling under uncertainty: Survey and research directions / T. Chaari [et al.] // Proceedings of Advanced Logistics and Transport (ICALT). – 2014. – P. 229-234.

4. Laburthe, F. Choco solver documentation / F. Laburthe, N. Jussien. – Nantes : Ecole de Mines de Nantes, 2011. – 224 p.

Система по подведению результатов матчей

М. В. Пывина

Студент магистрант

В. С. Тарасов

Ассистент

Введение

С каждым годом развитие игровой индустрия продолжает расти. Разработчики выпускают много интересных игр с красивой графикой и увлекательным сюжетом. Число пользователей растет с каждым днём. Из-за этого возникает необходимость в сборе данных и их анализе.

На данный момент возникает сложность, так как многие игры имеют собственный интерфейс со сложными шрифтами и крайне тяжело даже человеку правильно распознать никнейм пользователя. У многопользовательских игр нет открытого API, которые давали бы итоговые результаты, для дальнейшей аналитики. Сейчас подсчёт результатов матчей происходит в ручном режиме. Работа эта рутина, на нее тратиться большое количество человекочасов. Велик шанс ошибиться. Человек не машина и способен качественно выполнять

работу такого рода всего пару часов. Требуется автоматизация данного процесса.

1. Постановка задачи

Целью данной работы является разработать систему по подведению результатов матчей по скриншотам (снимкам экрана) из многопользовательских мобильных онлайн игр.

Система состоит из нескольких компонентов и представлена на рис. 1:

1. Сервис по распознаванию результатов матча;
    1.1. Модуль распознавания;
    1.2. Модуль сопоставления и принятия решения
2. Бот-кликер.

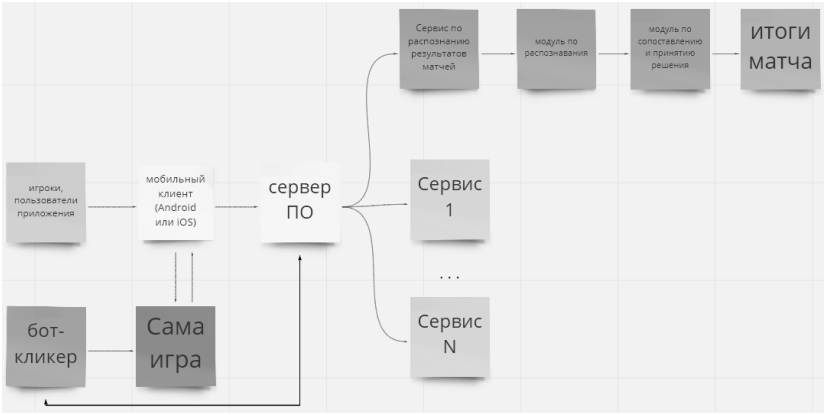


Рис. 1. Схематическое изображение основных компонентов системы

2. Анализ OCR технологий

Для анализа готовых OCR было выбрано 4 популярных сервиса:

1. Azure OCR [1];
2. Google OCR [2];
3. Yandex OCR [3];
4. Abby OCR [4].

Для начала было решено протестировать все четыре OCR на скане листа А4, скан содержал текст на английском языке с простым шрифтом. Ниже приводим таблицу результатов данного тестирования.

Таблица

Сравнение характеристик разных OCR

	Azure OCR	**Google OCR**	**Yandex OCR**	**Abby OCR**
Количество поддерживаемых языков	1 (Английский)	Около 60	35	Около 150-200
Распознавание рукописного текста	есть	есть	нет	есть
Размер загружаемого файла	4 мб	2 мб	1 мб	100 мб
Результат запроса	JSON	JSON	JSON	XML
Время распознавания базового текста (1 страница)	3 секунды	4 секунды	8 секунд	13 секунд
Вероятность совпадения содержания базового текста с результатом распознавания	99%	99%	99%	98%
Вероятность совпадения содержания сложного текста с результатом распознавания	48%	57%	51%	72%

3. Модуль распознавания

Модуль распознавания изображён на рис. 2, представляет собой сервис, получающий на вход снимки экрана из мобильных игр и выдающий на выход распознанную текстовую информацию из определенных областей снимка.

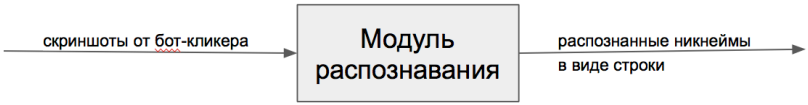


Рис. 2. Схематическое изображение работы модуля распознавания

4. Модуль сопоставления и принятия решений

Данный модуль представлен на рис. 3, отвечает за сопоставление ников, которые игроки вводят при входе в игровое лобби и ников из рейтинговой таблицы игры. Сложность состоит в том, что интерфейс игры предполагает разного типа шрифты – от самых простых и

понятных до сложных авторских. Поэтому важно сделать правильное сопоставление ников.

Модуль реализован на основе Алгоритма Ратклиффа/Обершелпа (Ratcliff/Obershelp pattern recognition).

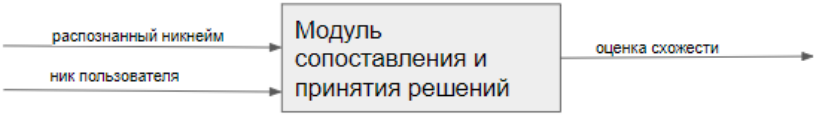


Рис. 3. Схематическое изображение работы модуля сопоставления и принятия решений

5. Attention-based модель нейросети

Для улучшения модуля распознавания была выбрана attention-based [5] модель нейросети. Она основана на сверточных нейронных сетях, рекуррентных нейронных сетях и новый механизме attention-based - который обеспечивает точность 84,2% на сложном датасете французских уличных знаков (FSNS).

Данная модель базируется на распознавание текста в неограниченной естественной среде является сложной задачей компьютерного зрения и машинного обучения. Традиционные системы оптического распознавания символов (OCR) в основном ориентированы на извлечение текста из отсканированных документов. Текст, полученный из естественных сцен, является более сложным из-за визуальных артефактов, таких как искажение, окклюзии, направленное размытие, загроможденный фон или различные точки зрения.

Разработчики модели утверждают, что найден компромисс между скоростью и точностью, который возникает в результате использования экстракторов признаков CNN различной глубины, что глубже не всегда лучше (с точки зрения точности, а также скорости). Данная модель простая, точная и быстрая, что позволяет использовать её для решения различных сложных задач извлечения текста из реального мира.

Новый механизм attention-based позволяет извлекать структурированную текстовую информацию, читая только интересные части всего изображения.

6. Итоговые результаты работы системы

На рис. 4 видно, что по никам получилась высокая вероятность совпадений. А с цифрами ещё предстоит работать в дальнейшем чтобы добиться лучшего результата. Так же заметны различия в работе модулей в зависимости от диагонали устройств.

Всего игроков: 42	Распознано	Не распознано	Распознано неверно	% Ошибок
Ник	42	0	0	0
Место	35	0	2	5,71%
Килы	42	0	0	0

Эмулятор

	Место		Киллы	
	Нейросеть	Реальное	Нейросеть	Реальное
ultCmeTanKa	6	6	2	2
BALAKLAVAÈG	5	9	1	1
chidaMANDA	11	11	3	3
ultFRONDZ	23	23	0	0
NoSxTekel	15	15	0	0
atlMiller	7	7	1	1
AIR×ALEXAND	14	14	0	0
skyENVIOUS	27	27	0	0
nick005	2	2	4	4
mrakCONSTAN	20	20	1	1
CD火SPLITEX	24	24	2	2
LeanBOBA	41	41	0	0
easSMILE	20	25	1	1

Всего игроков: 42	Распознано	Не распознано	Распознано неверно	% Ошибок
Ник	42	0	0	0%
Место	39	0	3	7%
Килы	37	0	5	11,90%

Планшет

	Место		Киллы	
	Нейросеть	Реальное	Нейросеть	Реальное
ultCmeTanKa	6	6	2	2
BALAKLAVAÈG	9	9	1	1
chidaMANDA	11	11	3	3
ultFRONDZ	23	23	1	0
NoSxTekel	15	15	0	0
atlMiller	7	7	1	1
AIR×ALEXAND	14	14	0	0
skyENVIOUS	27	27	0	0
nick005	2	2	4	4
mrakCONSTAN	20	20	1	1
CD火SPLITEX	24	24	2	2
LeanBOBA	41	41	1	0
easSMILE	25	25	1	1

Рис. 4. Результат работы системы

Заключение

Получилось разработать систему по подведению результатов матчей, сейчас она работает с некоторой погрешностью, добиться высокого процента совпадения – 98%, с помощью модуля распознавания на основе OCR-алгоритма под разрешение эмулятора.

В планах: улучшить работу модуля до 99% без OCR, посредством рендеринга NickInGame с нужным шрифтом и размером для дальнейшего сопоставления.

Список литературы

1. Computer Vision API - v2.0 [Электронный ресурс] : статья. – Режим доступа : https://westus.dev.cognitive.microsoft.com/docs/
2. Cloud Vision API [Электронный ресурс] : статья. – Режим доступа : https://cloud.google.com/vision/docs/
3. Оптическое распознавание текста (OCR) [Электронный ресурс] : статья. – Режим доступа : https://cloud.yandex.ru/docs/vision/concepts/ocr/
4. ABBYY Cloud OCR SDK) [Электронный ресурс] : статья. – Режим доступа : https://www.ocrsdk.com/documentation/code-samples/
5. Attention-based Extraction of Structured Information from Street View Imagery [Электронный ресурс] : статья. – Режим доступа : https://arxiv.org/PDF/1704.03549.pdf

Мониторинг аппаратных ресурсов и состояния клиент-серверных приложений

Е. И. Симонов

Студент магистрант

А. Л. Гавшин

Ассистент

Введение

После разработки и внедрения приложения необходимо поддерживать его работоспособность. Из-за наличия ошибок в реализации, повышенной пользовательской нагрузки, недостаточного количества выделенных аппаратных ресурсов или по одной из множества других причин приложение может начать работать некорректно. Разумеется, всегда есть способ узнать о сбоях в работе системы – получить сообщение от конечных пользователей. Но, во-первых, такие сообщения всегда приходят с задержкой после возникновения проблем, во-вторых, сбои в работе системы не всегда заметны в интерфейсе пользователя и не всегда проявляются однозначно. Поэтому, рассчитывая на сообщения от пользователей, можно столкнуться с тем, что система некоторое время будет находиться в состоянии некорректной работы. Это может приводить к отрицательным последствиям: от замедления работы приложения (и как следствие к оттоку пользователей) до временно неработающей системы (как следствие – к потери данных или финансовым потерям).

Разумеется, необходимо минимизировать время, в течение которого приложение работает некорректно. Для этого следует как можно быстрее начинать работы по устранению неисправностей и проводить эти работы как можно быстрее. Если время на устранение неисправности полностью зависит от компетенции разработчика (здесь и далее «разработчик» подразумевает инженера или разработчика программного обеспечения, системного администратора и т. д.), то время начала работ зависит от того, как быстро станет известно о сбое.

В данной работе рассматривается подход к реализации системы мониторинга приложения с целью решения вышеописанной проблемы. Описанный подход применим только к клиент-серверным приложениям в силу того, что в приложениях других типов затруднена возможность непрерывно отслеживать их состояние.

1. Цель работы

Простейший вариант реализации системы мониторинга – периодическая проверка работоспособности системы. Так как реализация только этой проверки является слишком простой и недостаточно полезной, к системе выдвигаются следующие функциональные требования:

- проверка работоспособности всех составных частей системы;
- возможность задания и изменения конфигураций и правил, согласно которым будет сделан вывод о работе приложения;
- выявление проблем, вероятность возникновения которых возрастает;
- визуализация текущего состояния системы;
- оповещение разработчиков о сбоях в работе системы.

2. Настройка составных частей приложения

Для начала необходимо настроить те составные части приложения, которые планируется мониторить. Для этого необходимо расширить их конфигурацию таким образом, чтобы они периодически предоставляли метрики, то есть численные значения, которые характеризуют различные параметры приложения (использование памяти, использование ЦПУ, количество HTTP запросов, количество потоков и т.д.).

В случае компонент, которые реализуют бизнес-логику, дополняются конфигурации самих компонент. Например, для приложений, написанных на Spring Boot, достаточно добавить конфигурации для Spring Boot Actuator [1], представленную в листинге 1 зависимость, и включить доступ к этим endpoint-ам как показано в листинге 2.

Листинг 1

Добавление Maven зависимости для настройки Spring Boot Actuator

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Листинг 2

Включение доступа к end-point-ам для получения метрик в файле свойств приложения

```
management.endpoints.web.exposure.include = *
```

После этого компонент будет предоставлять список метрик по запросу на их получение. Результат добавления этих конфигураций представлен на рис. 1.

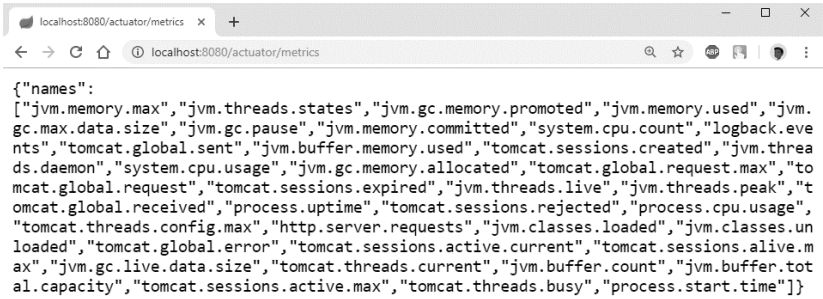


Рис. 1. Список метрик приложения по запросу

Для баз данных не нужно менять их конфигурации.

3. Настройка модуля по сбору метрик

Теперь необходимо воспользоваться инструментом, который будет с заданной периодичностью получать, сохранять и обрабатывать метрики. Одним из самых популярных инструментов для этих целей является Prometheus [2]. Поэтому рассмотрим его использование для поставленной задачи.

В большинстве баз данных не предусмотрен механизм предоставления метрик. Для этого необходимо установить так называемые экспортеры. Устанавливать их следует в той же среде, где установлена база данных. В качестве примера в листинге 3 приведены команды по установке и запуску экспортера для PostgeSQL. Так как большинство экспортеров написаны на языке Go, также следует предварительно установить его.

Листинг 3

Установка и запуск экспортера для PostgreSQL

```
$ go get github.com/wrouesnel/postgres_exporter
$ cd ${GOPATH-
$HOME/go}/src/github.com/wrouesnel/postgres_exporter
$ go run mage.go binary
$ export
DATA_SOURCE_NAME="postgresql://login:password@hostname:port/db
name"
$ ./postgres_exporter
```

Как было отмечено в предыдущем разделе, компоненты, реализующими бизнес-логику, уже могут предоставлять метрики. Но необходимо настроить взаимодействие между ними и Prometheus. Для этого, на примере того же Spring Boot, необходимо добавить зависимость, представленную в листинге 4.

Листинг 4

Добавление Maven зависимости для настройки взаимодействия с Prometheus

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

В результате Prometheus сможет считывать метрики с нового endpoint, представленного на рис. 2 (так как список довольно длинный, приведена лишь его часть).

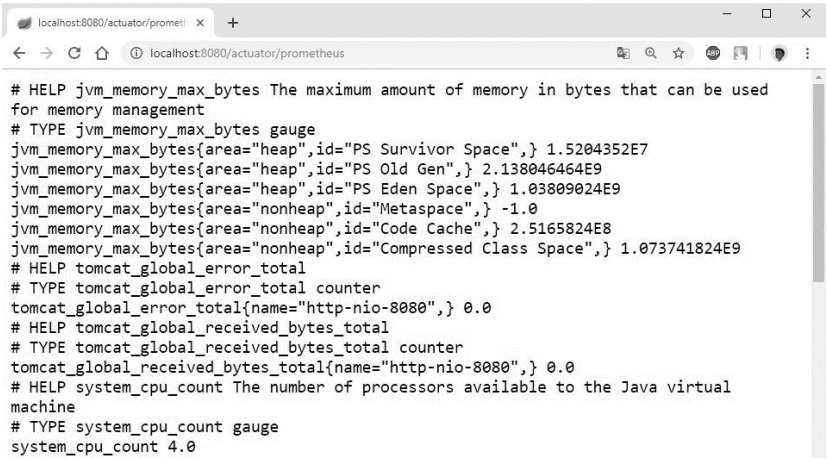


Рис. 2. Дополнительный end-point с метриками для Prometheus

Наконец следует сконфигурировать Prometheus. После установки необходимо изменить параметры *job_name: 'prometheus'* в соответствии с листингом 5, а затем запустить, выполнив команду в директории Prometheus: *prometheus --config.file=prometheus.yml*.

Листинг 5

Настройка Prometheus

```
- job_name: 'prometheus'
  metrics_path: '/actuator/prometheus'
  scrape_interval: 1s
  static_configs:
  - targets: ['APP_HOST:APP_PORT']
```

В результате уже на данном этапе можно получать значения метрик и отслеживать их на графике. Пример представлен на рис. 3.

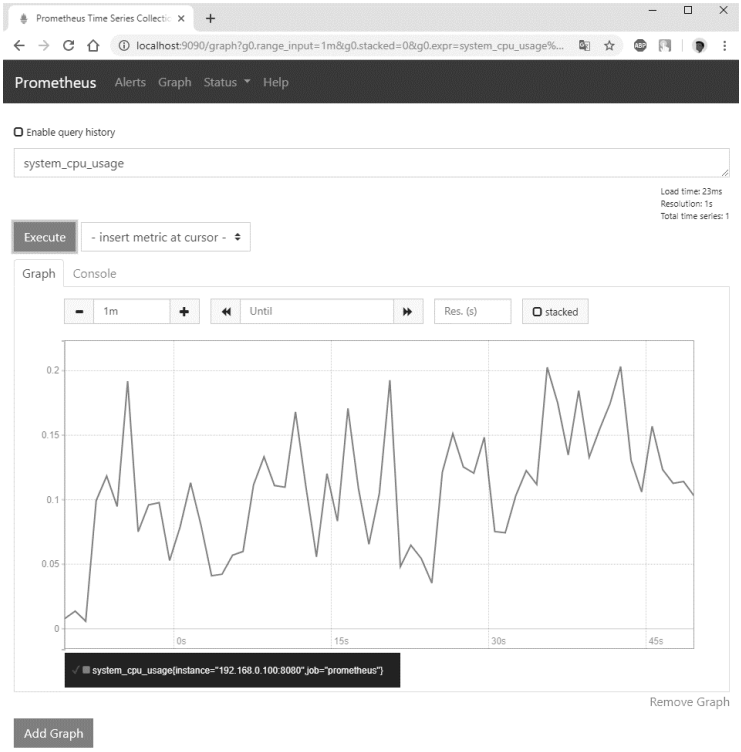


Рис. 3. Визуализации метрики использования ЦПУ с помощью Prometheus

4. Настройка модуля визуализации и оповещений

Как отмечалось ранее, текущие версии Prometheus поддерживают возможность визуализации данных. Но по удобству использования и настройки есть более подходящие средства. Например, Grafana, которая

позволяет в режиме реального времени визуализировать данные и отправлять оповещения в случае срабатывания заданного условия (например, если приложение начало использовать более 90% выделенной памяти).

Чтобы настроить Grafana, необходимо указать настроенный ранее Prometheus в качестве источника данных. Эта настройка осуществляется с помощью интерфейса пользователя и представлена на рис. 4.

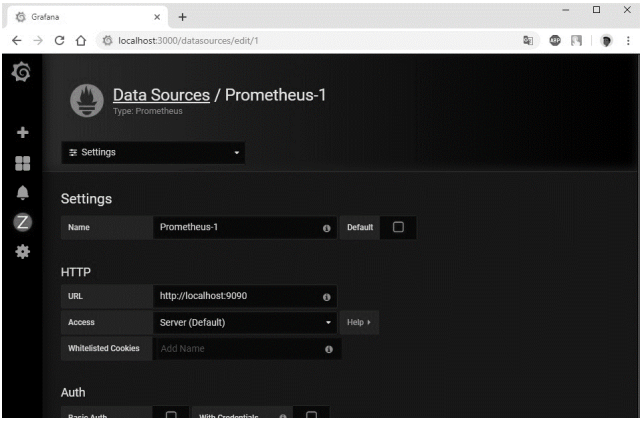


Рис. 4. Настройка взаимодействия Grafana с Prometheus

После настройки взаимодействия Grafana и Prometheus необходимо настроить так называемые доски с графиками и оповещения. Grafana предоставляет удобный интерфейса пользователя, и настройка будет не сложнее предыдущей. Поэтому она не приводится в данной статье. Пример настроенной доски приведен на рис. 5.

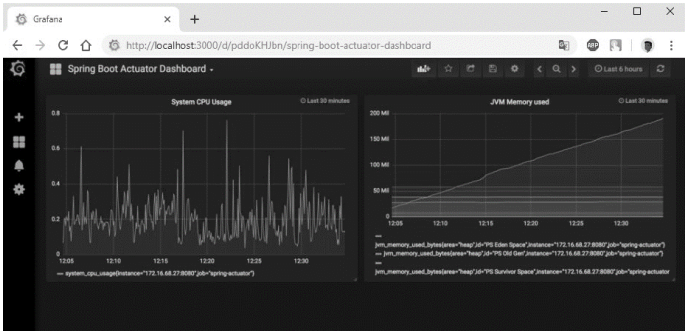


Рис. 5. Пример доски с графиками в Grafana

5. Дальнейшие улучшения

На данном этапе разработана система мониторинга, которая схематически изображена на рис. 6.

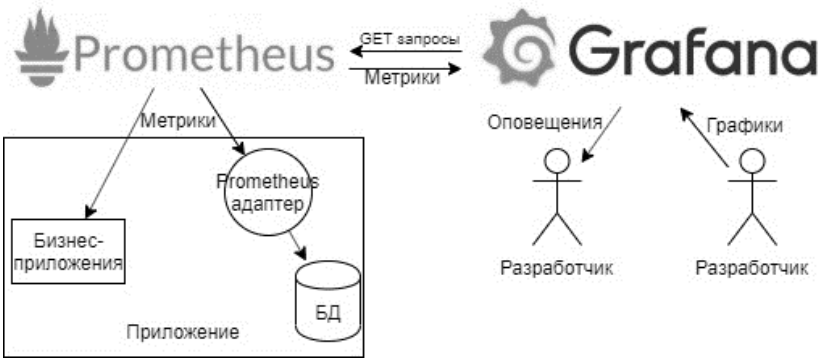


Рис. 6. Схема разработанной системы мониторинга

При желании можно улучшить реализацию рассматриваемой системы мониторинга, например, следующими способами:

1. Prometheus не предоставляет возможности хранить значения метрик долгое время. Для этих целей можно использовать так называемые базы данных временных рядов.
2. Для большей простоты все примеры настроек и установок были приведены для локальной машины localhost. Разумеется, лучше делать это с использованием Docker.
3. Для большей наглядности графиков можно писать свои метрики, которые вычисляются на основе нескольких других метрик.
4. Спроектировать набор правил, которые оповещают разработчика о том, что в ближайшее время возможно появление сбоев в работе приложения. И отправлять разработчикам оповещения заранее, уже перед появлением проблем в работе приложения.

Заключение

Данная статья посвящена разработке системы мониторинга состояния приложения. В ходе работы была разработана и реализована система для разработчиков, которая предлагает им отслеживать текущее состояние приложения и, самое главное, минимизирует время на оповещение о возникновении сбоев в работе приложения. Также были предложены способы улучшить разработанную систему, но их реализация выходит за рамки данной статьи.

Список литературы

1. Walls, C. Spring Boot in Action / C. Walls. – Shelter Island : Manning, 2016. – 266 p.
2. Prometheus [Электронный ресурс]. – Режим доступа : https://prometheus.io

Использование машинного обучения для разрешения кореференции

С. С. Соловьев

Студент магистрант

В. В. Гаршина

Доцент

Введение

В лингвистике под кореференцией традиционно понимают референциальное тождество имен. Задача разрешения тождественной референции заключается в том, чтобы соотнести упоминание в тексте какой-либо сущности с другими ее упоминаниями.

Выявление отношений кореференции весьма значимо при решении различных задач, связанных с автоматической обработкой естественного текста, так как обеспечивает возможность поиска имеющейся в анализируемом тексте информации об экстенсионалах имен.

За последние три десятилетия было проведено много исследований, связанных с задачей разрешения кореференции. Стоит отметить семантико-синтаксический анализатор SemSin, который с определенной точностью способен выявлять анафорические связи [1].

Однако подход машинного обучения для данной задачи стал применяться относительно недавно. Одна из первых работ по применению подхода машинного обучения к задаче разрешения кореференции была описана в статье Soon W.M. [2]. Автором была введена парная модель (mention-pair model) разрешения кореференции, которая с тех пор широко используется.

Эта модель работает следующим образом: для каждой именной фразы, которая может быть упоминанием, генерируется ряд возможных антецедентов из предыдущих именных фраз. Для каждой пары вызывается классификатор. Выбирается первая (или лучшая, в

зависимости от алгоритма) положительная пара. Аналогично создается набор пар для обучения классификатора.

Несмотря на то, что эта модель имеет ряд недостатков, например она допускает несовместимые пары в цепочке, модель все еще широко используется, особенно в качестве базовой.

После того как RuCor, впервые открытый русский корпус с аннотацией кореференции, появилась возможность создать систему разрешения кореференции для русского языка.

1. Терминология. Типы кореференции.

Антецедент – предшествующие упоминание, на которое ссылается местоимение. Пример представлен на рис. 1

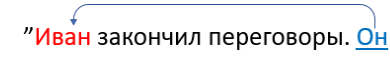


Рис. 1. Разновидность текстового отношения «антецедент»

Анафо́р – ссылающееся слово (здесь: личное местоимение).

Референт – истинный объект действия (Иванов Иван – сотрудник компании.)

Кореференция – совпадение какого-либо сегмента текста с референтом.

Различают следующие типы кореференции.

1. Грамматическая кореференция:
   - дейктическая (местоименная)
   - возвратные местоимения (себе, свой…)
   - личные местоимения (чаще 3-го лица: она, они, её, их)
   - относительное местоимение который
   - подчинительные местоимения и союзы (кто, что, где, когда)
   - указательные местоимения (этот, тот)
   - имена собственные
   - именные группы (дескрипции)

В случае грамматической кореференции антецедент высчитывается на основе грамматических правил языка. Грамматическая кореференция практически никогда не переходит границ предложения, ее всегда можно представить как отсылку одного узла к другому, следовательно ее аннотирование легче автоматизировать.

2. Прономинальная текстовая кореференция:
   - личные и притяжательные
   - местоимения третьего лица
   - указательное местоимение этот
   - синонимы:

– транслитерация: Yandex – Яндекс
– аббревиация: ВТБ – Внешторгбанк – Банк Внешней Торговли
– синонимы: больница – госпиталь
– словообразование: Москва – московский
– графические: авто кредит – автокредит
– титулы: премьер- министр, папа римский

Отсылка реализуется не только за счет грамматических средств языка, но и на основании знания контекста. Текстовая кореференция может легко переходить границы предложения.

2. Модели разрешения кореференции с использованием машинного обучения.

Используются три основных класса моделей разрешения кореференции: попарная модель (mention-pair model), модель ранжирования (ranking model) и сущностная модель (entity-based model).

Попарная модель. Попарная модель является классификатором, который определяет, являются ли упоминания в паре кореферентными. Решения о кореферентности каждой пары принимаюся независимо друг от друга. Наиболее распространёнными алгоритмами кластеризации являются closest-first и best-first.

Сущностная модель. В отличие от попарной модели, сущностная модель классифицирует пары упоминаний, используя информацию из уже созданных (в т. ч. частично) кластеров. Данный подход позволяет получать больше информации для принятия решения, которой часто бывает недостаточно при попарном сравнении. Более того, эта модель поддерживает транзитивность, не требуя постобработки.

Модель ранжирования. Модель ранжирования считается промежуточным шагом между попарной и сущностными моделями. При использовании данной модели на одном шаге может быть проверено более одного упоминания-кандидата, таким образом проводится соревнование между всеми кандидатами и выбирается наиболее подходящий.

3. Основные составляющие проектируемой нейросети.

В данной статье для обучения нейросети используется способ обучения с учителем, базовая схема которого представлена на рис. 2.

Наблюдаемые величины (наблюдения, observations) – сущности, поведение которых мы хотим предсказать. В нашем случае это пары упоминаний, связь между которыми нужно установить.

Целевые переменные (targets) – соответствующие наблюдаемым величинам метки. Обычно именно их мы и предсказываем.

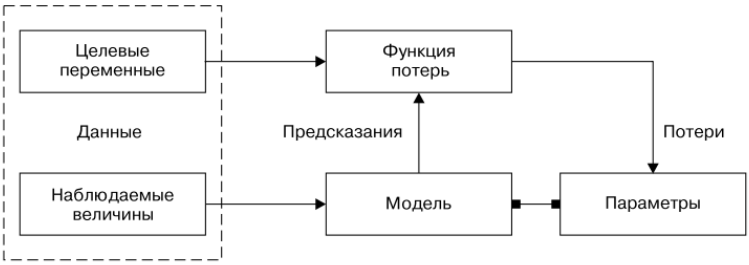


Рис. 2. Основные элементы проектируемой нейросети

Модель – представляет собой математическое выражение или функцию, принимающую на входе наблюдаемую величину и предсказывающую значение целевой метки.

Параметры – иногда называются также весами и служат для параметризации модели.

Предсказания, называемые также оценками (estimates), представляют собой значения целевых переменных, предсказанные моделью по наблюдаемым величинам.

Функция потерь (loss.function) – это функция, служащая мерой отклонения предсказания от целевой переменной для наблюдений из обучающей последовательности. Функция потерь ставит целевой переменной и ее предсказанию в соответствие скалярное вещественное значение, называемое потерями (loss).Чем меньше значение потерь, тем лучше модель предсказывает целевую переменную.

В качестве исходных данных мы используем размеченный корпус RuCor для русского языка. Для моделирования нейросети корпус исходных данных разделяется на две группы. Одна для тренировки нейросети (train), а другая для тестирования (test).

Изначально был создан набор классификаторов пар упоминаний с использованием простых неглубоких признаков, предложенный в статье Soon [2], система, которая часто используется в качестве основы для моделей машинного обучения для разрешения кореференции.

Некоторые особенности, используемые автором в статье, неприменимы к русскому языку в простой форме, например признак определенного существительного (если существительное начинается с определенного артикля).

На вход нейросеть (рис. 3) будет получать параметрический вектор, описанный ниже.

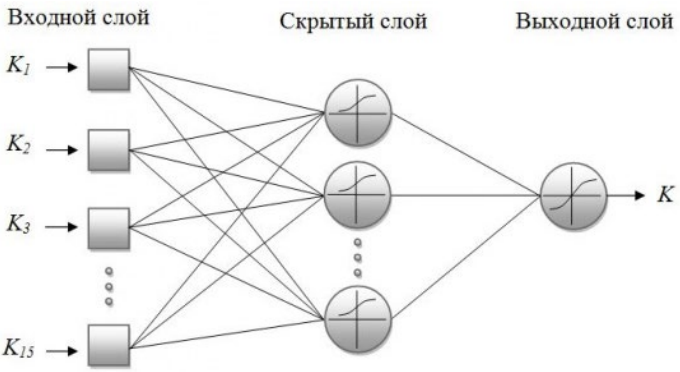


Рис. 3. Архитектура нейронной сети

Чтобы компенсировать это, в базовой системе мы заменили эти функции на следующие:

– Соглашение об одушевленности: верно, если *i* и *j* одушевлены неодушевлены.

– Совпадения строк *i* и *j* после удаления указательных местоимений. (Если j и j не местоимения); где *i* – потециальный антецедент, *j* – анафора.

Таким образом, были выделены следующие признаки для классификатора нейросети:

– Расстояние между *i* и *j* составляет 1 предложение.
– Совпадения строк *i* и *j* после удаления указательных местоимений. (Если *i* и *j* не местоимения)
– Согласованность *i* и *j* в одушевленности.
– *j* – это местоимение.
– *i* – это местоимение.
– *i* и *j* являются местоимениями.
– Гендерное соответствие *i* и *j*.
– *i* и *j* совпадают в числе.
– *i* и *j* – имена собственные.
– *j* – указательное местоимение
– *i* и *j* находятся в аппозиционной связи.

Параметрический вектор, полученный для каждой пары упоминаний, подается на вход нейросети, а на выходе возвращается 1 (true), если в паре упоминаний присутствует кореферентная связь и 0 (false), если таковая отсутствует.

Заключение

Представленная модель позволяет получить неплохие результаты для распознавания кореферентных связей с точностью до 60%. Для улучшения результатов можно настраивать данную модель путем изменения удаления/добавления составляющих параметрического вектора. При этом нельзя получить однозначную универсальную модель, подходящую под все типы кореференции. Этот момент хорошо раскрыт в статье С. Толдовой о влиянии семантических признаков на разрешение кореференции в русском языке [3]. Так же следует понимать, что точность нейросети зависит от размера обучаемой выборки – чем больше корпус данных для обучения, тем будут выше показатели точности нейросети. Необходимо иметь в виду и переобучение нейронной сети, когда построенная модель хорошо работает на примерах их обучающей выборки, но относительно плохо показывает себя на примерах, не участвовавших в обучении.

Список литературы

1. Боярский, К.К. Выявление анафорических отношений при автоматическом анализе текста / К. К. Боярский, Е. А. Каневский, А. В. Степукова // Научно-технический вестник информационных технологий, механики и оптики. – 2013. – №5. – С. 108-112.

2. Soon, W. M. A machine learning approach to coreference resolution of noun phrases / W. M. Soon, D. C. Y. Lim, H. T. Ng // Computational Linguistics. – 2001. – Vol. 27. – №4. – P. 521-544.

3. Толдова, С. Разрешение кореферентных связей для русского языка: вклад семантических признаков. / С. Толдова // Компьютерная лингвистика и интеллектуальные технологии. – 2017. – Т. 1. – С. 339-349.

Исследование архитектуры DSP процессора в системе на кристалле 1892ВМ14Я компании ЭЛВИС

В. М. Солодухин

Студент магистрант

Д. Н. Борисов

Доцент

Введение

Компания Элвис одна из немногих кто активно разрабатывает и внедряет микросхемы на территории России. Предприятие было основано в марте 1990 года на базе структурного подразделения научно-производственного объединения «ЭЛАС». В свое время они вели разработки в области космической электронной техники, в том числе разработку собственных САПР, а также полностью законченные аппаратно-программные бортовые системы управления и обработки информации космического базирования серий «Салют», в частности, функционировавших на борту станции «МИР».

Помимо этого, компания также занимается разработкой микросхем типа «Систем на кристалле» (или сокращенно СнК) на базе собственной платформы проектирования «Мультикор».

Среди них:

– процессоры «Мультикор»;
– радиационно-стойкие микросхемы для космических аппаратов;
– микросхемы для СВЧ трактов широкополосных систем связи.

Процессоры серии «Мультикор» – это однокристальные программируемые многопроцессорные «системы на кристалле» созданные на базе библиотеки IP-ядер [1] платформы Мультикор.

Процессоры данной серии сочетают в себе качества двух классов приборов: цифровых процессоров обработки сигналов и микроконтроллеров [2], что с учетом ограниченного размера позволяет решать одновременно две задачи: контроля и высокоточной обработки информации [3].

В данной работе рассматривается архитектура DSP кластера в системе семейства «Мультикор» 1892ВМ14Я [4], проводится обзор системы команд, самой структуры кластера, а также вычислительных возможностей ядер.

1. Описание системы и её назначение

Микропроцессор «Мультиком-02» (MCom-02, 1892ВМ14Я) [4] впервые был продемонстрирован публике в 2016 году. Данный процессор является 6-ядерным сигнальным микропроцессором с пониженным энергопотреблением, созданным для связных, мультимедийных, навигационных, а также встраиваемых мобильных приложений, таких как, например: планшетов, интеллектуальных видеокамер, телефонов.

Микросхема изготовлена по КМОП-технологии с минимальными топологическими размерами элементов 40 нм. Процессор 1892ВМ14Я [4] имеет в своем распоряжении различные контроллеры памяти и может поддерживать такие типы памяти как DDR3/ SRAM/ PSRAM/ ROM/ NOR FLASH/ NAND FLASH/ SD/ MMC. Помимо этого, система имеет на своем борту различный набор интерфейсов: Ethernet MAC 10/100/1000, UART, USB 2.0 (HOST+DEVICE+PHY), I2C, SPI, I2S.

Основные же модули, выполняющие основную работу данной системы представлены следующим списком:

– стандартный управляющий процессор представленный сдвоенным процессорным ядром – Dual CORTEX-A9 (CPU 0-1) с FPU-акселератором и NEON SIMD-акселератором (ARM);

– DSP-кластер на основе двух DSP-ядер ELcore-30M с полной программной совместимостью с микросхемами 1892ВМ10Я и 1892ВМ15АФ;

– графический блок, состоящий из 2D/3D акселератор (MALI-300, ARM) с поддержкой OpenVG 1.1 и OpenGL ES 2.0/1.1, а также с поддержкой разрешения до HD 1080p;

– видеокодек VELcore-01, обеспечивающий функции H.264 CBP Encode and Decode, Full HD (1920x1080) стерео поток с частотой следования не менее 30 кадров/с; память видеоданных VRAM объемом 1 Мбайт, доступная для CPU и DSP;

На рис. 1 изображена схема модуля и его составных частей. Данная система выпускается разработчиком в виде двух отладочных наборов: Салют-ЭЛ24Д1 и Салют-ЭЛ24Д2.

По своей архитектуре, данный микропроцессор находит свое активное применение в системах машинного зрения, в том числе и интеллектуальных видеокамерах; обработке видео и аудиопотоков с расчетом на сложные производственные условия.

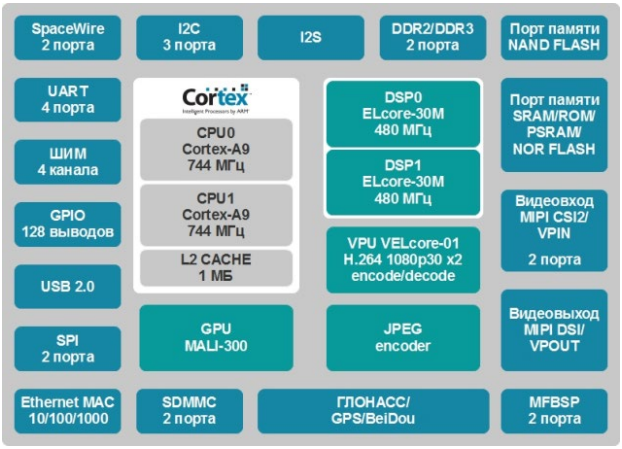


Рис. 1. Схема основных модулей

2. Структура DSP-кластера

Одним из центральных элементов в данной микросхеме является DSP-кластер DELcore-30M [5], который представлен двумя DSP-ядрами сопроцессорами-акселераторами сигнальной обработки.

Само ядро имеет гарвардскую архитектуру с внутренним параллелизмом по потокам обрабатываемых данных, основное предназначение которого обработка информации в форматах с фиксированной и с плавающей точкой. Система инструкций позволяет реализовать параллельно несколько вычислительных операций и пересылок. Ядро имеет 7-фазный программный конвейер и гибкие адресные режимы, что позволяет также реализовывать алгоритмы сигнальной обработки с высокой производительностью. Каждое DSP-ядро управляется CPU-ядром.

Как было сказано выше, наличие внутреннего параллелизма добавляет больше возможностей для обработки данных в 8/16/32/64-разрядных форматах с фиксированной точкой, 32-разрядном формате с плавающей точкой в стандарте IEEE754, либо программно в формате с плавающей точкой 32E16, тем самым, давая возможность выбирать между точностью и производительностью.

Пиковая суммарная производительность DSP-кластера имеет следующие значения для разных операций:

– в формате плавающей точки: 16 операций с плавающей точкой (IEEE 754) за 1 такт;

– в формате фиксированной точки (int32): 16 32-битных операций с фиксированной точкой за 1 такт;

– в формате фиксированной точки (int16): 64 16-битных операций с фиксированной точкой за 1 такт;

– в формате фиксированной точки (int8): 96 байтовых операций с фиксированной точкой за 1 такт.

Внешний интерфейс DSP-кластера реализуется с помощью шины CDB, в то время как доступ к программной памяти осуществляется по интерфейсу AXI Switch, позволяющий передавать по 64 бита за такт. Для каждого ядра кластера предусмотрен собственный сигнал синхронизации, позволяющий отключать тактовую частоту от каждого ядра с целью снижения энергопотребления.

Кроме того, каждое ядро в кластере обладает собственной программной памятью, позволяя каждому ядру работать независимо. Для синхронизации работы в кластере используется два механизма: механизм прерываний и механизм обменов через XBUF [6]. При этом ядра могут формировать прерывания для друг друга. То есть ядро, которое получило прерывание, переходит в режим RUN и начинает выполнение подпрограммы, адрес которой храниться в специальном регистре этого ядра.

Память в кластере организована в виде системы с ассиметричным доступом к памяти (NUMA). Вся память разбита на 2 сегмента и для каждого ядра есть свой сегмент, обращение к которому не приводит к простою самого ядра. При этом каждое из ядер имеет свою программную память объемом 32 кбайт и общую для всех память объемом 256 кбайт. Такой подход дает возможность одновременного доступа к памяти программ и данных DSP как со стороны центрального процессора, так и со стороны сигнального процессора. Блоки общей и программной памяти реализованы в виде 2-х портовых элементов.

Все DSP ядра имеют возможность обращаться к ресурсам процессора, в связи с чем, адресация внутренней памяти DSP кластера не изменена, что позволяет использовать их совместимость.

3. Система инструкций

Система инструкций DSP-ядра Elcore-30M имеет в наличии базовую часть и расширение, которое позволяет работать с 64/128 разрядными данными. На рис. 2 приведены обозначения, принятые при описании расширения системы инструкций.

Базовая система инструкций позволяет работать с данными, формат которых не должен превышать 32 разряда. Особенностью системы инструкций DSP-ядра Elcore-30M [5] заключается в ориентированности на высокопроизводительную параллельную обработку данных, что является одним из самых важных аспектов в быстродействии.

Обозначение регистра	Разрядность регистра	Номера	Варианты форматов данных
T /S / D	16	31:0	R = r[15:0]
T.L / S.L / D.L	32	31:0	L = (l_1 , l_0), l_1 = L[31:16], l_0 = L[15:0] L = (complex) LX = l_1 + j l_0
T.D / S.D / D.D	64	31:0	D = (d_3 , d_2 , d_1 , d_0), D_3 = D[63:48], d_2 = D[47:32], d_1 = D[31:16], d_0 = D[15:0] D = (LX_1 , LX_0), LX_1 = d_3 + j d_2, LX_0 = d_1 + j d_0 D = (complex) DX = L_1 + j L_0, L_1 = (d_3 , d_2), L_0 = (d_1 , d_0)
T.Q / S.Q / D.Q	128	30:0 16 четных	Q = (q_7, q_6, q_5, q_4, q_3, q_2, q_1, q_0), q_7 = D[127:112], q_6 = D[111:96], q_5 = D[95:80], q_4 = D[79:64], q_3 = D[63:48], q_2 = D[47:32], q_1 = D[31:16], q_0 = D[15:0] Q = (DX_1, DX_0) = (LX_3, LX_2, LX_1, LX_0), LX_3 = q_7 + j q_6, LX_2 = q_5 + j q_4, LX_1 = q_3 + j q_2, LX_0 = q_1 + j q_0 Q = (L_3 , L_2 , L_1 , L_0), L_3 = (d_7 , d_6), L_2 = (d_5 , d_4), L_1 = (d_3 , d_2), L_0 = (d_1 , d_0)

Рис. 2. Форматы данных в регистровом файле

В этом случае инструкции размещаются в программной памяти DSP-ядра последовательно в порядке возрастания адреса, причем в рамках одной инструкции может выполняться несколько команд.

Каждая инструкция занимает одно или два 32-разрядных слова. Адрес чтения программной памяти формируется с помощью регистра программного счетчика PC, автоматически инкрементирующегося (на 1 или 2) при последовательном ходе программы. Команды (операции) DSP-ядра по своему действию делятся на три больших группы:

- вычислительные команды;
- команды пересылок;
- команды программного управления.

Вычислительные команды производят действия над данными, которые хранятся в регистровом файле DSP-ядра, а полученные результаты помещаются обратно в этот же регистр. Вычислительные команды, в свою очередь, делятся по характеру исполняемой операции и по форматам обрабатываемых данных на более мелкие группы команд (команды сложения, вычитания, умножения). Значительная часть операций использует вычислительные команды.

Команды пересылок используются для обмена данными между регистрами, регистрами и памятью или при загрузке непосредственных данных в регистры.

Команды программного управления производят изменения в последовательности исполнения инструкций DSP-ядра. С их помощью организуются программные переходы, циклы, входы в подпрограмму и выходы из нее, остановка DSP-ядер.

На выполнение инструкций накладывается ряд ограничений:

– одновременно исполняемые вычислительные операции и пересылки не должны иметь одинаковые адреса операндов-приемников;

– количество повторений цикла DO не должно превышать 16383;

– количество вложенных циклов не должно превышать семи, что связано с глубиной стека циклов.

Все инструкции выполняются на программном конвейере, который содержит 7 фаз. Конвейеризация используется для возможности обработки нескольких инструкций на разных стадиях. Если отсутствуют торможения конвейера, то скорость выполнения инструкций в конвейерном режиме составляет одну инструкцию в течение всего цикла.

Время исполнения операций на конвейере отличается в зависимости от типа. Для всех логических операций, операций транзита, загрузки аккумулятора время выполнения составляет 1 такт. Для остальных операций, время исполнения составляет 2 такта.

Заключение

В данной работе была рассмотрена архитектура DSP-кластера системы на кристалле 1892ВМ14Я [4] компании ЭЛВИС. В результате исследования, можно сделать вывод, что процессор обладает достаточной мощностью и необходимым набором инструментов для решения задач в области обработки аудио и видео данных, из-за чего находит активное применение в системах машинного зрения. Быстродействие процессора достигается за счет высокой скорости выполнения команд с помощью конвейеризации и высокой степени параллелизма, достигаемой реализацией гарвардской архитектуры в ядре с внутренним параллелизмом по потокам обрабатываемых данных. Одним из плюсов также можно считать низкое энергопотребление, осуществляемое как на аппаратном, так и на программном уровне.

Список литературы

1. Baer, J. Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors / J. Baer. – Cambridge : Cambridge University Press, 2009. – 382 p.

2. Таненбаум, Э. Архитектура компьютера / Э. Таненбаум, Т. Остин. – Питер : Санкт-Петербург, 2019. – 816 с.
3. Иванова, В. Цифровая обработка сигналов и сигнальные процессоры [Электронный ресурс] : учебное пособие / В. Иванова, А. Тяжев. – Самара : Поволжский государственный университет телекоммуникаций и информатики, 2017. – 253 с.
4. Микросхема интегральная 1892ВМ14Я [Электронный ресурс] : Руководство пользователя. – Режим доступа : https://multicore.ru/mc/data_sheets/Manual_1892VM14YA.pdf
5. DSP-кластер DELcore-30M. Архитектура [Электронный ресурс] : документация – Режим доступа : https://multicore.ru/mc/data_sheets/Manual_DELcore-30M_031210.pdf
6. Уильямс, Г. Отладка микропроцессорных систем / Г. Уильямс. – М. : Энергоатомиздат, 1988. – 253 с.

Практическое применения метода анализа иерархий в подборе сотрудников в судебную систему

В. А. Соляников

Студент магистрант

Е. Ю. Митрофанова

Доцент

Введение

В современном мире происходит интенсивное внедрение компьютерных технологий во все сферы жизнедеятельности человека, начиная с образования и заканчивая космической отраслью. Нет ни одной области, где бы не использовались ЭВМ, а также разработанное программное обеспечение для решения той или иной задачи. В данной статье пойдет речь о способе применения метода анализа иерархий в судебной системе Российской федерации, с помощью которой можно в будущем решить задачу по подбору лучшего кандидата для должности в судебном аппарате нашей страны. Для этого будет подробно описан механизм практического применения метода анализа иерархий с конечным результатом.

1. Метод анализа иерархий

Суть метода анализа иерархий (МАИ) заключается в том, чтобы сложную проблему разложить на её простые составляющие части. Затем осуществить обработку суждений лицом, принимающим решения (ЛПР), на основе парных сравнений.

Этапы применения МАИ [1-2]:

1. определение цели (проблемы), критериев и альтернатив для построения иерархического дерева;
2. построение иерархического дерева;
3. сформировать приоритеты критериев и оценить каждую из альтернатив по критериям, определив наиболее важную их них;
4. рассчитать вектор приоритетов;
5. определение согласованности приоритетов;
6. корректировка суждений;
7. иерархический синтез.

Этап №1. Определение цели (проблемы), критериев и альтернатив для построения иерархического дерева.

В качестве глобальной цели выбрана задача выбора подходящего кандидата на должность помощника судьи. Цель всегда находится на вершине иерархии (первый уровень).

На втором уровне иерархии определяются критерии, которые будут способствовать нахождению цели. Для данной задачи сформировано четыре критерия: институт получения юридического образования, юридический стаж, возраст, знания иностранных языков.

На третьем уровне указывается количество альтернатив (в данной задаче их количество равно 4).

Для удобства применения полученной информации, отобразим ее в табл. 1.

Этап №2. Построение иерархического дерева.

На основе вышеизложенных данных можно построить иерархическое дерево с выявленной целью, критериями и альтернативами (рис. 1).

После построения иерархии происходит установление приоритетов для элементов всех уровней путём парных сравнений. На следующем этапе формируется набор локальных приоритетов и осуществляется синтез глобальных приоритетов [2].

Этап №3. Определение приоритетов критериев и оценивание каждой из альтернатив по критериям, определив наиболее важную из них.

Таблица 1

Исходные данные

Альтернативы		**Критерии**			
		Юридическое образование	Юридический стаж	Возраст	Знание иностранных языков
		C1	C2	C3	C4
Антоненко С.В.	A1	МГУ	5 лет	27 лет	English
Петренко Ю.П.	A2	ТГУ	10 лет	32 года	English and Deutsch
Азимутов Р.К.	A3	БГУ	6 лет	28 лет	Не владею
Щербаков Ю.С.	A4	ВГЛТУ	3 года	25 лет	Deutsch

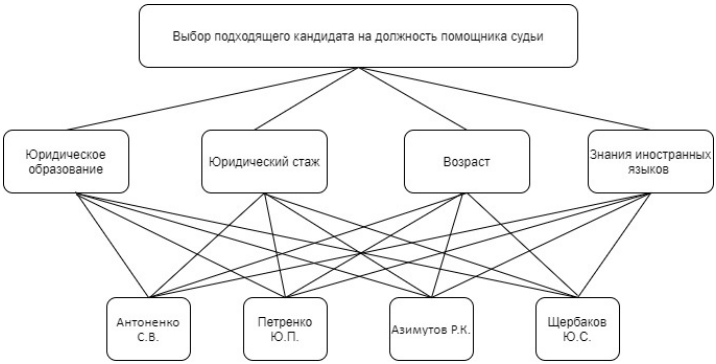


Рис. 1. Иерархическое дерево

Для удобства определения приоритетов критериев и оценивания каждой из альтернатив по критериям мы будем использовать программу Excel. Составим матрицу сравнения критериев (рис. 2).

Матрица критериев		Знания иностранных языков	Юридическое образование	Юридический стаж	Возраст	Находим произведение каждого критерия (построчно)	Извлекаем √ из произведения и находим собствен-ный вектор	Нормализованные оценки вектора приоритета
		C1	C2	C3	C4			
Знания иностранных языков	C1	1	1/8	1/7	1/2	0,0089	0,3074	0,0531
Юр. Образование	C2	8	1	1	5	40	2,5149	0,4346
Юр. Стаж	C3	7	1	1	5	35	2,4323	0,4203
Возраст	C4	2	1/5	1/5	1	0,0800	0,5318	0,0919
						Сумма:	5,7864	1

Рис. 2. Матрица сравнения критериев

У матрицы критериев по главной диагонали – единицы, т.к. будет осуществляться пересечения одинаковых критериев. Далее оценивается каждый из критериев по шкале отношений. В данном примере используется девятибалльная шкала (степени значимости действий). Далее необходимо найти собственный вектор.

Для нахождения произведения оценок параметров критерия «знания иностранных языков», необходимо выполнить перемножение строчных значений 1*1/8*1/7*1/2, в результате получается величина 0,0089. Затем аналогичным образом выполняются соответствующие величины для каждого критерия.

Полученное произведение для каждого из критериев возводится в степень равную 1/количество критериев. Для этого используется следующая формула:

$$K_1 = \sqrt[n]{p_1} = p_1^{1/n}. \quad (1)$$

Данное значение будет являться собственным вектором критерия «знания иностранных языков».

Далее необходимо определить сумму векторов критериев для того, чтобы найти нормализованные оценки вектора приоритета [3].

Для нахождения суммы векторов критериев, необходимо сложить все показатели собственных векторов критериев по формуле:

$$S = \sum_{i=1}^{n} K_n. \quad (2)$$

А для того, чтобы найти нормализованные оценки вектора приоритета, необходимо собственный вектор критерия / сумму векторов критериев.

Чтобы оценить правильность полученных данных необходимо определить величины ОС (однородность суждений) и ИО (индекс однородности) (рис. 3).

Первоначально вычисляется сумма по столбцам по каждому критерию. Далее находится произведение суммы по столбцам и нормализованной оценки вектора приоритета. Для получения суммы

λmax необходимо сложить все произведения суммы по столбцам и нормализованную оценку вектора приоритета [4].

	Знания иностранных языков	Юридическое образование	Юридический стаж	Возраст	Сумма λmax
	C1	C2	C3	C4	
Сумма по стобцам	18,00	2,33	2,34	11,50	
Произведение суммы по столбцам и нормализованной оценки вектора приоритета	0,956225481	1,010485924	0,984816083	1,056969974	4,008497462
ОС	0,3	< 10%			
ИС	0,0028				

Рис. 3. Однородность суждений, индекс однородности

Далее нам необходимо найти ИС (индекс согласованности) по формуле:

$$ИС = \frac{\lambda_{\max} - n}{n-1} \quad (3)$$

ОС (однородность суждений) определяется по формуле:

$$ОС = \frac{ИС}{ИС_r} * 100\%, \quad (4)$$

где $ИС_r$ – индекс согласованности при случайной оценке сравнений.

Если ОС меньше 10%, то критерии оценены объективно.

Далее проводится подобный анализ относительно всех представленных критериев (см. рис. 4-11).

Знания иностранных языков	A1	A2	A3	A4	Находим произведение каждого критерия (построчно)	Извлекаем √ из произведения и находим собствен-ный вектор	Нормализованные оценки вектора приоритета
A1	**1**	1/6	1	1/2	0,08	0,5373	0,0916
A2	6	**1**	8	5	240,00	3,9360	0,6713
A3	1	1/8	**1**	1	0,1250	0,5946	0,1014
A4	2	1/5	1	**1**	0,4000	0,7953	0,1356
					Сумма:	**5,8631**	**1**

Рис. 4. Матрица сравнения критерия «Знания иностранных языков»

Сумма по стобцам	A1	A2	A3	A4	Сумма Lmax
	10,00	1,49	11,00	7,50	
Произведение суммы по столбцам и нормализованной оценки вектора приоритета	0,916378	1,00137	1,11555	1,017293	4,05059305
ОС	1,9	< 10%			
ИС	0,0169				

Рис. 5. Однородность суждений, индекс однородности критерия «Знания иностранных языков»

Юридическое образование	A1	A2	A3	A4	Находим произведение каждого критерия (построчно)	Извлекаем √ из произведения и находим собствен-ный вектор	Нормализованные оценки вектора приоритета
A1	1	4	5	9	180	3,6628	0,6335
A2	1/4	1	3	2	1,5000	1,1067	0,1914
A3	1/5	1/3	1	2	0,1333	0,6043	0,1045
A4	1/9	1/2	1/2	1	0,0278	0,4082	0,0706
					Сумма:	5,7820	1

Рис. 6. Матрица сравнения критерия «Юридическое образование»

Сумма по стобцам	A1	A2	A3	A4	Сумма Lmax
	1,56	5,83	9,50	14,00	
Произведение суммы по столбцам и нормализованной оценки вектора приоритета	0,9889	1,1165	0,99283	0,9885	4,086759805
ОС	3,2	< 10%			
ИС	0,0289				

Рис. 7. Однородность суждений, индекс однородности критерия «Юридическое образование»

Юридический стаж	A1	A2	A3	A4	Находим произведение каждого критерия (построчно)	Извлекаем √ из произведения и находим собствен-ный вектор	Нормализованные оценки вектора приоритета
A1	1	1/4	1/2	1	0,125	0,5946	0,1028
A2	4	1	5	8	160	3,5566	0,6151
A3	2	1/5	1	3	1,2	1,0466	0,1810
A4	1	1/8	1/3	1	0,041666667	0,4518	0,0781
					Сумма:	5,6496	1

Рис. 8. Матрица сравнения критерия «Юридический стаж»

Сумма по стобцам	A1	A2	A3	A4	Сумма Lmax
	8,00	1,58	6,83	13,00	
Произведение суммы по столбцам и нормализованной оценки вектора приоритета	0,8227	0,96879	1,23693	1,0158	4,044212933
ОС	1,6	< 10%			
ИС	0,0147				

Рис. 9. Однородность суждений, индекс однородности критерия «Юридический стаж»

Возраст	A1	A2	A3	A4	Находим произведение каждого критерия (построчно)	Извлекаем √ из произведения и находим собствен-ный вектор	Нормализованные оценки вектора приоритета
A1	1	1/7	1	1	0,1429	0,6148	0,1063
A2	7	1	4	6	168	3,6002	0,6227
A3	1	1/4	1	4	1	1,0000	0,1729
A4	1	1/6	1/4	1	0,0417	0,4518	0,0781
					Сумма:	5,6668	1

Рис. 10. Матрица сравнения критерия «Возраст»

Сумма по стобцам	A1	A2	A3	A4	Сумма Lmax
	10,00	1,56	6,25	12,00	
Произведение суммы по столбцам и нормализованной оценки вектора приоритета	1,0633	0,97104	1,08093	0,9377	4,052907383
ОС	2,0	< 10%			
ИС	0,0176				

Рис. 11. Однородность суждений, индекс однородности критерия «Возраст»

Используя полученные данные, необходимо вычислить вектор глобального приоритета (рис. 12).

ГЛОБАЛЬНЫЕ ПРИОРИТЕТЫ ВЫБОРА						
АЛЬТЕРНАТИВЫ		КРИТЕРИИ				Глобальный приоритет
		Знания иностранных языков	Юридическое образование	Юридический стаж	Возраст	
		C1	C2	C3	C4	
		Численное значение вектора приоритета				
		0,0531	0,4346	0,4203	0,0919	
Антоненко Сергей	A1	0,0916	0,6335	0,1028	0,1063	0,3331915636
Петренко Юлия	A2	0,6713	0,1914	0,6151	0,6227	**0,4346341644**
Азаматов Руслан	A3	0,1014	0,1045	0,1810	0,1729	0,1427938934
Щербаков Юрий	A4	0,1356	0,0706	0,0781	0,0781	0,0779195129

Рис. 12. Глобальные приоритеты выбора

В результате, чтобы получить матрицу с векторами глобальных приоритетов, необходимо указать численное значение вектора приоритета. Для этого используются данные из матрицы критериев «Нормализованные оценки вектора приоритета», а также нормализованные векторы из каждого рассматриваемого критерия [5].

Проанализировав данные, можно сделать вывод, что наилучшим кандидатом на должность помощника судьи будет человек под альтернативой A2 с наибольшим значением глобального приоритета.

Заключение

В данной статье описан один из способов принятия решений, а именно метод анализа иерархий. МАИ позволяет за кратчайшие сроки выяснить необходимые данные для принятия решения в сложных проблемных ситуациях. Этот метод применяется для выбора наиболее подходящего кандидата на должность помощника судьи в судебной системе Российской Федерации. На основе данной статьи можно разработать программное обеспечение для работников кадрового подразделения судебной системы разных инстанций, что облегчит подбор высококвалифицированных сотрудников в будущем.

Список литературы

1. Андрейчиков, А. В. Анализ, синтез, планирование решений в экономике : учебник / А. В. Андрейчиков, О. Н. Андрейчикова. – 2–е изд., доп. и перераб. – М. : Финансы и статистика, 2004. – 464 с.

2. Саати, Т. Принятие решений. Метод анализа иерархий / Т. Саати. – М. : Радио и связь, 1993. – 278 с.

3. Дэвид, Г. Метод парных сравнений / Г. Дэвид ; под ред. Ю. Адлер. – М. : Статистика, 1978. – 134 с.

4. Макаров, И. М. Теория выбора и принятия решений / И. М. Макаров. – М. : Наука, 1982. – 327 с.

5. Микони, С. В. Теория и практика рационального выбора / С. В. Микони. – М. : Маршрут, 2004. – 462 с.

Алгоритм многоуровневой полигональной сегментации растровых изображений

В. В. Сукманов

Студент магистрант

В. В. Фертиков

Доцент

Введение

В настоящее время цифровая обработка, анализ и распознавание изображений являются одними из наиболее интенсивно развивающихся направлений научных исследований, это обусловлено широким распространением и практическим использованием систем видеонаблюдения, моделирования графических объектов, машинного зрения в различных технических отраслях и науке.

Одной из важнейших технологий данных научных исследований, называемой сегментацией, является процесс разбиения изображения на отдельные части, содержащие в себе пиксели, объединённые по некоторым признакам. Целью такого разбиения, называемого сегментацией, является изменение и/или упрощение представления цифрового изображения для дальнейшего анализа. В статье предлагается алгоритм автоматической сегментации, работающий по известному методу морфологических водоразделов [1] с расстановкой маркеров. Специфика сегментируемых алгоритмом объектов заключается в наличии прямолинейных участков в их контурах, что фактически означает определенную «правильность» формы.

1. Метод морфологических водоразделов

Рассмотрим изображение как некоторую топографическую карту местности, где значения яркостей представляют собой значения высот относительно уровня моря. При постепенном заполнении местности водой образуются бассейны, затем, при наполнении, бассейны начинают объединяться. Место такого соприкосновения называется линией водораздела, чтобы вода из разных бассейнов не смешивалась, здесь ставится барьер. Процедура наполнения продолжается до тех пор, пока все высоты не окажутся под водой. Результатом такой операции будут разделённые барьерами и залитые водой области. Данный алгоритм называется методом морфологических водоразделов [1].

Существуют две вариации алгоритма водоразделов, которые отличны целью, условием применения, а также реализацией. Самым простым способом построения линий водораздела является использование морфологической дилатации. В нем решается задача выделения на однородном фоне изображения однородных по яркости объектов (пятен), таким образом, метод зачастую применяется не к самому изображению, а к градиенту этого изображения. Но присутствие шумов на градиентном изображении создаёт большое количество локальных минимумов, что приводит к избыточной сегментации, которая может быть настолько значительной, что сделает результат практически бесполезным. Решением проблемы является ограничение допустимого количества областей путём добавления априорных знаний – маркеров. Именно данный подход использован описываемой в статье разработкой. Маркер представляет собой отметку на изображении, которая достоверно относит помеченную область к одному из сегментов или к фону. Обычно, процедура расстановки маркеров выполняется в интерактивном режиме, однако достоинством предлагаемого алгоритма является функция автоматической расстановки маркеров без участия оператора.

2. Подготовка контурного препарата

Целью заявленного алгоритма является выделение на изображении объектов специального вида: контур каждого из них имеет как минимум один прямолинейный участок. Само изображение содержит шумы, неравномерные засветки фона и объектов, также присутствует вложенность одних объектов в другие. Пример такого изображения представлен на рис. 1.

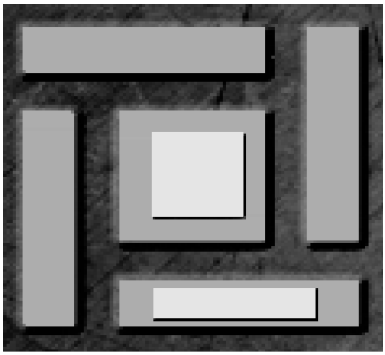


Рис. 1. Пример вложенности объектов

Также алгоритм должен обладать невысокой вычислительной сложностью, а разработанное приложение – гибкостью настроек.

Далее описана последовательность действий, предусмотренная разработанным алгоритмом многоуровневой сегментации изображений с автоматической расстановкой маркеров. Алгоритм реализован в компьютерной программе при помощи библиотеки алгоритмов компьютерного зрения OpenCV и языка программирования Java [2].

Последовательность шагов обработки следующая. Входное изображение анализируется детектором границ Кэнни [1] для выявления контурных точек, с предварительно применённым фильтром Гаусса, для снижения уровня шума. Результатом анализа будет т.н. контурный препарат, который на следующем шаге обрабатывается методом трансформации Хафа [3], для обнаружения точек, лежащих на прямой линии. Таким образом, конечным результатом преобразования является набор отрезков, которые состоят из точек, сгруппированных по степени близости к прямым линиям. В реализации эти подходы имеют множество параметров и гибкую настройку, позволяющую задавать допуски на искривление отрезков контура, разрывы, длину итогового отрезка.

3. Кластеризация пикселей маркерной маски

После получения набора отрезков начинается расстановка маркеров вдоль каждого отрезка по обеим его сторонам. На этом этапе все расставленные маркеры принадлежат к единственному общему типу. После окончания процедуры расстановки по координатам каждого пикселя каждого маркера извлекается яркость соответствующих пикселей из оригинального изображения и заносится в т.н. маркерную маску. Сегмент такой маски представлен на рис. 2.

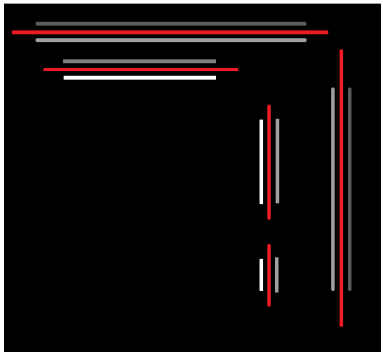


Рис. 2. Сегмент маркерной маски

В следующем шаге алгоритма пиксели, содержащиеся в полученной маркерной маске, подвергаются кластеризации методом k-средних [2]. Целью данного метода является разделение m наблюдений из пространства R_n на заданное k кластеров, при этом каждое наблюдение относится к тому кластеру, к центроиду которого оно ближе всего. В нашем случае все пиксели маркерной маски являются исследуемыми объектами и описываются как наблюдения X в пространстве R.

$$X^{(j)} = \left(X^{(1)}, X^{(2)}, ..., X^{(m)}\right), \quad X^j \in R_n$$

Так как наше изображение в градациях серого, то яркость каждого пикселя характеризуется числом от 0 до 255. Таким образом, расчёт расстояния от каждого центра до каждого пикселя, т.е. мера близости рассчитывается по формуле Евклидового расстояния

$$\sqrt{\left(B_1 - B_2\right)^2},$$

где B – яркость пикселя.

После завершения кластеризации появляется информация о центрах кластеров, т.е. яркости пикселей маркерной маски разделяются на несколько доминирующих.

4. Анализ маркеров и принятие решений

Зная доминирующие яркости маркерной маски, мы можем анализировать каждый маркер в отдельности. Для этого находим среднее значение яркости всех пикселей маркера и определяем, к какому центроиду кластера этот маркер ближе всего расположен. Для удобства и быстроты работы алгоритма, все созданные маркеры были занесены в специальную структуру данных – карту-хранилище, в которой ключом является суммарная яркость пикселей, а значением – набор маркеров.

Одновременно с определением подходящего кластера для маркера, автоматически определяется и тип маркера – фон, внешний объект или внутренний объект, однако, если суммарная яркость расположена далеко от центров кластеров, то такой маркер отбрасывается как ненадёжный. Порог принятия такого решения задаётся пользователем.

На рис. 3 показан результат работы алгоритма над изображением, представленным на рис. 1.

5. Тестирование и ближайшие цели

Предварительное тестирование разработанной программы производилось как на искусственных, так и на реальных изображениях, взятых из хранилища университета Беркли [4] (The University of

California, Berkeley). Некоторые результаты тестов представлены на рис. 4 и на рис. 5.

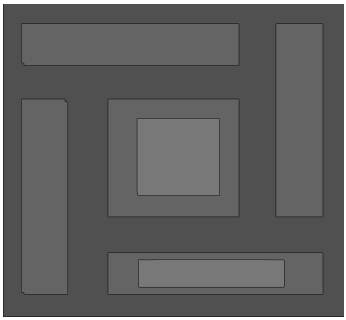
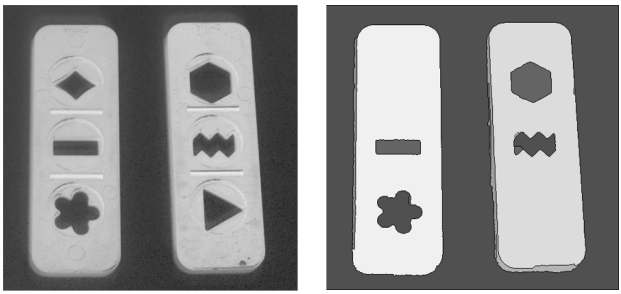


Рис. 3. Результат работы алгоритма



а б

Рис. 4. Пример сегментации: а – исходное изображение, б – сегментированное изображение

Данный подход обладает некоторыми недостатками. Одним из таких недостатков является необходимость заранее указывать количество нужных кластеров, что не всегда удобно и в некоторых случаях приводит к результатам, не удовлетворяющим оператора. Ко всему прочему, возможна недостаточно удачная для последующего использования расстановка начальных центров кластеров. Все эти недостатки в данный момент преодолеваются, и в ближайших целях планируется уход от фиксированного указания кластеров путем комплексирования алгоритма к-средних с результатами порогового анализа гистограммы распределения яркости пикселей маркеров. Эти доработки приведут к полной автоматизации расстановки маркеров, представленного в данной статье алгоритма.

а

б

в

Рис. 5. Сегментация с различным количеством кластеров: а – исходное изображение, б – пять кластеров, в – четыре кластера

Заключение

Данная статья посвящена разработке и реализации алгоритма многоуровневой полигональной сегментации растровых изображений. Описана последовательность операций по созданию предварительного контурного препарата, автоматической расстановки маркеров и последующий анализ маркерной маски, для принятия решения о надёжности того или иного маркера.

Алгоритм реализован в виде рабочей программы, снабженной демонстрационным пользовательским интерфейсом. Использование данного решения позволяет гибко настраивать представленный алгоритм, добавлять новые операции, получать наглядные результаты после каждого шага. Всё это позволяет проводить анализ данных и улучшать алгоритм.

Список литературы

1. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1072 с.

2. Прохоренок, Н. OpenCv и Java. Обработка изображений и компьютерное зрение / Н. Прохоренок. – Санкт-Петербург : БХВ-Петербург, 2018. – 320 с.
3. Прэтт, У. Цифровая обработка изображений. / У. Прэтт. – Москва : Мир, 1982. – 310 с.
4. The Berkeley Segmentation Dataset and Benchmark [Электронный ресурс] : база данных. – Режим доступа : https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

BigData-анализ веб-пользователей в маркетинге

И. М. Трусов

Студент магистрант

А. С. Коваль

Старший преподаватель

Введение

Большие Данные (англ., BigData), на сегодняшний момент, являются одним из ключевых драйверов развития информационных технологий и имеют обширную сферу использования, так как традиционные методы анализа информации уже не могут угнаться за внушительными объемами постоянно растущих и обновляемых данных. Актуальность данной работы заключается в том, что на сегодняшний день в связи с повсеместным и динамичным развитием и ростом интернет-торговли, современные условия экономической конкуренции требуют создания все новых и новых технологий привлечения покупателей и увеличения продаж.

1. Системы поддержки принятия решений

Обширным классом задач, которые необходимо решать с использованием Больших Данных, являются задачи сетевого маркетинга и, в частности, построение современных рекомендательных систем.

Рекомендательными системами называют класс систем поддержки принятия решений, которые используют информацию о предпочтениях человека для рекомендации ему контента, товара или услуг о которых он ранее не знал, но которые могут оказаться полезными или интересными для него [1].

Рекомендательные системы обычно подбирают список рекомендаций одним из двух способов: методом коллаборативной фильтрации или фильтрацией на основе содержания (контентной). Суть метода контентной фильтрации состоит в том, что он использует характеристики объектов, оцененных пользователем ранее, и затем рекомендует пользователю объекты с похожими свойствами. Коллаборативная же фильтрация строит модель исходя из истории поведения пользователя и составляет рекомендации на основе решений, принимаемых пользователями с похожей историей.

Каждый из рассмотренных базовых подходов имеет свои преимущества и недостатки. К достоинствам метода коллаборативной фильтрации можно отнести получение более разнообразных и точных рекомендации, но существует проблема «холодного старта» из-за разреженности данных. Что касается контентной фильтрации, к ее преимуществам можно отнести отсутствие проблемы «холодного старта» для контента, также для ее работы не нужна обширная база пользователей, так как метод не требует от них оценок. Но в рекомендациях, полученных с помощью контентной фильтрации нет эффектов новизны, неожиданности, разнообразия, а также, чтобы вынести рекомендацию нужно спрогнозировать оценки для всех товаров (долгий и ресурсоемкий процесс).

Таким образом, основной целью данной работы является реализация гибридного метода для построения системы поддержи принятия решений, который позволил бы в той или иной мере избежать недостатков каждого из подходов и в конечном результате мог повысить эффективность и точность рекомендательных систем. Так же были реализованы вышеописанные базовые методы фильтрации, для того чтоб провести сравнение.

Для демонстрации наилучшего результата работы алгоритмов мною был использован открытый набор данных службы рекомендаций по фильмам, описывающий 5-звездочный рейтинг и содержащий около 100 тысяч оценок для 9000 фильмов [2]. Для построения рекомендательных систем был выбран язык Python, который широко используется в аналитике и является одним из основных средств для построения моделей машинного обучения [3].

2. Реализация контентной фильтрации

Для фильтрации на основе содержимого мы сначала получаем элементы, в которых заинтересован целевой пользователь, для каждого из этих элементов находим другие элементы, имеющие сходные характеристики, таким образом создается исчерпывающий список. В начале работы, система узнает у пользователя о его любимых фильмах и

на основании этого получает следующий данные о средней оценке всех фильмов данного списка, а также их жанров.

После того, как система собрала все жанры, интересующие целевого пользователя, она получает все фильмы, которые относятся к любому из этих жанров. Предположим, что набор жанров, который интересует пользователя, – это A. Для всех фильмов, извлеченных на этом шаге, определяется жанр каждого из этих фильмов, назовем этот набор B. Далее сортируется список по убыванию, таким образом, что фильм со значением B∩A будет сверху. После сортировки данных, мы извлечем 50 лучших фильмов. Эти данные будут использованы для применения алгоритма k ближайших соседей. Лучшие k фильмов, найденные ближе всего к фильмам, оцененных пользователем, будут рекомендованы целевому пользователю.

3. Реализация коллаборативной фильтрации

Коллаборативная фильтрация для своей работы использует данные пользователей, интересы которых совпадают с интересами целевого пользователя. Если сходство будет велико, то можно предположить, что интересы пользователей практически одинаковы, и, таким образом Система составляет для целевого пользователя рекомендации на основе решений, принимаемых пользователями с похожей историей.

Для работы алгоритма мы используем формулу коэффициента Пирсона. Предположим, у нас есть n пользователей. Каждый из этих пользователей просмотрел несколько фильмов и оценил их. Согласно описанию, приведенному в наборе данных, каждый пользователь оценил по меньшей мере 20 фильмов.

Возьмем двух пользователей X и Y. Необходимо рассчитать коэффициент Пирсона для этих двух пользователей для того, чтобы понять, насколько коррелированы данные. Для начала необходимо будет извлечь набор общих фильмов, которые видны и оценены как по пользователю X, так и по пользователю Y. Это будет пересечение X∩Y. Для любого фильма по пересечению по X у нас есть рейтинг X и Y. Теперь, мы рассчитаем коэффициент Пирсона по следующей формуле:

$$r(X,Y) = \frac{\sum_k (X_k - \bar{X})(Y_k - \bar{Y})}{\sqrt{\sum_k (X_k - \bar{X})^2 \sum_k (Y_k - \bar{Y})^2}}, \quad (1)$$

где $r(X,Y)$ – коэффициент Пирсона между X и Y, X_k,Y_k – оценки элемента k пользователями X и Y, а $\bar{X}$, $\bar{Y}$ – среднее значение оценок X и Y. После расчета коэффициента Пирсона для целевого пользователя по отношению ко всем остальным пользователям мы будем хранить эти данные в некоторой внутренней структуре данных, далее рассматриваем

несколько лучших результатов (пользователей, для которых существует существенная корреляция и высокое значение пересечения). Создается исчерпывающий список фильмов, который не виден целевому пользователю, и рейтинг прогноза для всех этих фильмов рассчитывается по следующей формуле:

Теперь, мы рассчитаем коэффициент Пирсона по формуле:

$$p(X_i) = \frac{\sum_k Y_i - r(X,Y)}{n}, \quad (2)$$

где $p(X_i)$ – предсказанный рейтинг элемента *i* для пользователя *X*, а *Y* состоит из всех n людей, которые оценили элемент *i*. Таким образом, создается окончательный список, в котором сохранены названия фильмов и их прогнозируемые оценки для целевого пользователя. Этот список отсортирован в порядке убывания по отношению к прогнозируемым рейтингам, и отображаются первые несколько результатов.

4. Реализация гибридизированной фильтрации

Суть же гибридного метода в том, что он состоит из модуля, в основе которого лежат все вышеописанные базовые подходы, сгибридизированные между собой.

Работа данного алгоритма состоит из следующих шагов:

1. Применяется вышеописанный метод коллаборативной фильтрации, чтобы найти пользователей, с которыми целевой пользователь тесно связан
2. Система находит исчерпывающий набор жанров, который охватывает все фильмы, оцененные целевым пользователем
3. После того, как связанные пользователи будут найдены, мы узнаем фильмы, которые видят эти пользователи, но не видят целевой пользователь
4. Полученные данные хранятся в структуре данных, содержащей название фильма, его оценку и жанровое совпадение с текущим пользователем
5. Применяем метод k ближайших соседей, чтобы получить несколько лучших результатов (как и в методе контентной фильтрации)

5. Сравнение реализованных алгоритмов

Для проверки эффективности реализованных систем выбран следующий метод тестирования: из всех известных связей пользователь-фильм убиралось M фильмов, а после система для пользователя из каждой убранной связи должна была порекомендовать K фильмов. Реализованные системы сравнивались между собой при различных K

Точность алгоритма определяется с помощью метрики качества ранжирования Mean Average Precision (MAP), которая для своей работы в свою очередь использует метрику Average Precision (AP). AP высчитывает среднюю точность K рекомендаций для каждого пользователя, а MAP в свою очередь считает метрику ранжирования average precision для всех N пользователей и усредняет значение [4].

$$map@K = \frac{1}{N}\sum_{j=1}^{N} ap@K_j \tag{3}$$

где $ap@K = \frac{1}{K}\sum_{k=1}^{K} r^{\text{true}}(\pi^{-1}(k))\, p@k$, N – общее число пользователей, K – рекомендованные фильмы, $p@k$ – базовая метрика качества ранжирования для одного объекта.

В таблице приведено сравнение реализованных методов.

Таблица

Сравнение точности реализованных алгоритмов с помощью метрики MAP

	K=10	**K=15**	**K=20**
Коллаборативная	0.399	0.501	0.5472
Контентная	0.141	0.155	0.1845
Гибридизированная	0.420	0.525	0.5565

Даже несмотря на то, что контентная фильтрация показывает результаты намного хуже коллаборативной, реализованная гибридная система улучшила общий результат.

Заключение

Рекомендательные системы являются обширным классом моделей поддержки принятия решений, которые могут помочь практически каждому бизнесу. Реализовать РС для своего бизнеса можно с помощью различных подходов, рассмотренных в данном докладе. Но результаты экспериментального исследования показали, что гибридный подход является более динамичным и дает лучшие результаты, чем тот случай, когда оба других подхода применяются индивидуально, и что реализованная гибридизация действительно способна помочь улучшить результаты рекомендательных систем, причем даже относительно плохо работающая система может помочь улучшить результат хорошо работающей системы при корректном использовании гибридизации.

Список литературы

1. Рекомендательные системы [Электронный ресурс]. – Режим доступа : http://businessdataanalytics.ru/download/ RecommendationSystems.pdf
2. Набор данных MovieLens [Электронный ресурс] : база данных. – Режим доступа : https://grouplens.org/datasets/movielens/
3. Введение в Pandas: анализ данных на Python [Электронный ресурс] : обучающий курс. – Режим доступа : https://khashtamov.com/ru/pandas-introduction/
4. Метрики качества ранжирования [Электронный ресурс] : it-форум. – Режим доступа : https://habr.com/ru/company/econtenta/blog/303458/

О вырождениях орбит разложимых 7-мерных алгебр Ли

Д. И. Хацкевич

Студент магистрант

Г. В. Зинькевич

Студент магистрант

А. В. Лобода

Профессор

1. Основные математические термины, используемые в статье

Линейное пространство – множество, в котором определены операции сложения и умножения на скаляр [1].

Алгебра Ли (g) – линейное пространство, с дополнительной операцией $[e_1,e_2]$, называемой коммутатором, удовлетворяющее следующим условиям:

1. антикоммутативность $[e_1,e_2]=-[e_2,e_1]$;
2. тождество Якоби $[[e_1,e_2],e_3]+[[e_2,e_3],e_1]+[[e_3,e_1],e_2]=0$.

Размерностью Алгебры Ли ($\dim(g)$) называется количество векторов в любом базисе рассматриваемой алгебры.

Краткая запись векторного поля в J^4 в виде набора его компонентов:

$e = (a(z),(b(z),c(z),d(z)) = a(z)\frac{\partial}{\partial z_1} + b(z)\frac{\partial}{\partial z_2} + c(z)\frac{\partial}{\partial z_3} + d(z)\frac{\partial}{\partial z_4}$, где $a(z), b(z), c(z), d(z)$ – аналитические функции четырех комплексных переменных z_1, z_2, z_3, z_4.

Выпрямлением поля называется отображение, превращающее его в поле параллельных векторов одинаковой длины.

Теорема о выпрямлении векторного поля:

Всякое гладкое векторное поле локально выпрямляемо в окрестности каждой неособой точки (точки, где вектор поля отличен от нуля).

Любая искомая орбита (интегральная гиперповерхность M алгебры g_7) описывается в пространстве $\mathbb{J}^4$ вещественным уравнением $\Phi(z_1, z_2, z_3, z_4) = 0$ и удовлетворяет системе дифференциальных уравнений в частных производных:

$$\mathrm{Re}(e_k(\Phi)_{|M}) \equiv 0, k = \overline{1,7}, \quad (1)$$

означающей касание гиперповерхностью M любого из базисных полей алгебры g_7.

Гиперповерхность в пространстве $\mathbb{J}^4$ вырождена, если ее определяющая функция не зависит от одной или нескольких переменных этого пространства.

2. Постановка задачи и основной результат

В качестве начальной постановки задачи предлагалось найти в 4-мерном комплексном пространстве все невырожденные орбиты вещественных 7-мерных алгебр Ли векторных полей со структурой $g = g_5 + g_2$, где g_2 – нетривиальная двумерная алгебра Ли с соотношением $[e_1, e_2] = e_1$, а g_5 – любая из 5-мерных алгебр Ли, представленная своими коммутационными соотношениями на рисунке.

Отметим, что 5-мерные алгебры Ли из данной таблицы образуют блок алгебр, каждая из которых содержит 4-мерную абелеву подалгебру $< e_1, e_2, e_3, e_4 >$. Это означает, что базисные элементы этой подалгебры удовлетворяют соотношениям:

$$[e_i, e_j] = 0, i, j = \overline{1,4}. \quad (2)$$

Теорема. Пусть 7-мерная алгебра Ли g векторных полей в $\mathbb{J}^4$ имеет структуру $g = g_5 + g_2$, где g_5 – любая из 5-мерных алгебр Ли,

представленная на рисунке. Тогда все 7-мерные орбиты (интегральные поверхности) такой алгебры являются вырожденными.

Алгебры	$[e_1,e_2]$	$[e_1,e_3]$	$[e_1,e_4]$	$[e_1,e_5]$	$[e_2,e_3]$	$[e_2,e_4]$	$[e_2,e_5]$	$[e_3,e_4]$	$[e_3,e_5]$	$[e_4,e_5]$
$\mathfrak{g}_{5,7}$				e_1			αe_2		βe_3	γe_4
$\mathfrak{g}_{5,8}$							e_1		e_3	γe_4
$\mathfrak{g}_{5,9}$				e_1			e_1+e_2		βe_3	γe_4
$\mathfrak{g}_{5,10}$							e_1		e_2	e_4
$\mathfrak{g}_{5,11}$				e_1			e_1+e_2		e_2+e_3	γe_4
$\mathfrak{g}_{5,12}$				e_1			e_1+e_2		e_2+e_3	e_3+e_4
$\mathfrak{g}_{5,13}$				e_1			γe_2		pe_3-se_4	se_3+pe_4
$\mathfrak{g}_{5,14}$							e_1		pe_3-e_4	e_3+pe_4
$\mathfrak{g}_{5,15}$				e_1			e_1+e_2		γe_3	$e_3+\gamma e_4$
$\mathfrak{g}_{5,16}$				e_1			e_1+e_2		pe_3-se_4	se_3+pe_4
$\mathfrak{g}_{5,17}$				pe_1-e_2			e_1+pe_2		qe_3-se_4	se_3+qe_4
$\mathfrak{g}_{5,18}$				pe_1-e_2			e_1+pe_2		$e_1+pe_3-e_4$	$e_2+e_3-pe_4$

Рисунок. Коммутационные соотношения 5-мерной Алгебры Ли

3. Лемма об упрощении базисов

Так как алгебры g_5 и g_2 образуют прямую сумму $g=g_5+g_2$, то к указанным четырем векторным полям e_1,e_2,e_3,e_4 можно добавить еще одно из векторных полей алгебры g_2, например e_7. Новая 5-мерная подалгебра $<e_1,e_2,e_3,e_4,e_7>$ 7-мерной алгебры g также является абелевой.

В [2] предложена схема совместного упрощения (выпрямления) в 3-мерном комплексном пространстве базисных векторных полей алгебры Ли, содержащей абелеву подалгебру. Применяя технику этой работы в нашем случае 4-мерного комплексного пространства, можно получить следующее утверждение.

Лемма. Пусть g – 7-мерная алгебра Ли векторных полей в 4-мерном комплексном пространстве. Пусть, кроме того, g содержит 5-мерную абелеву подалгебру $h=<e_1,e_2,e_3,e_4,e_7>$ и имеет хотя бы одну невырожденную 7-мерную орбиту M в $\mathbb{J}^4$. Тогда аналитическими заменами координат z_1,z_2,z_3,z_4, определенными вблизи фиксированной точки $Q\in M$, можно привести базис h к одному из следующих 4-х видов:

$e_1:(1,0,0,0)$, $e_2:(0,1,0,0)$, $e_3:(0,0,1,0)$, $e_4:(0,0,0,1)$, $e_7:(iA,iB,iC,iD)$, или $e_1:(1,0,0,0)$, $e_2:(0,1,0,0)$, $e_3:(0,0,1,0)$, $e_4:(a_4(z_4),\ b_4(z_4),\ c_4(z_4),0)$, $e_7:(a_7(z_4),\ b_7(z_4),\ c_7(z_4),0)$, или $e_1:(1,0,0,0)$, $e_2:(0,1,0,0)$, $e_3:(a_3,b_3,0,0)$, $e_4:(0,0,1,0)$, $e_7:(0,0,0,1)$, или $e_1:(1,0,0,0)$, $e_2:(0,1,0,0)$, $e_3:(0,0,1,0)$, $e_4:(a_4(z_4),\ b_4(z_4),\ e_7:(0,0,0,1)$, где A,B,C,D – вещественные константы, $a_k(z_4)$, $b_k(z_4)$, $c_k(z_4)$ – функции одной комплексной переменной, $a_3(z_3,z_4)$, $b_3(z_3,z_4)$ – функции двух комплексных переменных.

4. Изучение четырех отдельных случаев

Рассмотрим первый тип базиса:

$$e_1:(1,0,0,0),e_2:(0,1,0,0),e_3:(0,0,1,0),e_4:(0,0,0,1),e_7:(iA,iB,iC,iD)$$

Проведем следующее преобразование такого вида базиса. Домножим первое векторное поле e_1 на вещественную константу A, второе поле – на B, третье и четвертое – на C и D, соответственно.

Так как алгебра Ли – это линейное пространство, следовательно, можно складывать векторные поля с вещественными коэффициентами, оставаясь в алгебре: $e_7^*:(A,0,0,0)+(0,B,0,0)+(0,0,C,0)+(0,0,0,D)=$ $=(A,B,C,D)\in g$.

Получаем в алгебре g в дополнение к e_7 параллельное ему поле e_7^*, составленное из вещественных констант: $e_7:(iA,iB,iC,iD),e_7^*:(A,B,C,D)$.

Линейной заменой координат превратим поле e_7^* в дифференцирование по новой переменной z_4^*, т. е. в $(0,0,0,1)$. Тогда e_7 перейдет в $(0,0,0,i)$.

При подстановке полей e_7 и e_7^* в систему дифференциальных уравнений (1) получим, что уравнение $\Phi(z_1,z_2,z_3,z_4)=0$, описывающее интегральную гиперповерхность M алгебры g_7, не зависит от новой комплексной переменной z_4, что означает ее вырождение.

Рассмотрим второй тип базиса:

$$e_1:(1,0,0,0),e_2:(0,1,0,0),e_3:(a_3(z_3,z_4),b_3(z_3,z_4),0,0),e_4:(0,0,1,0),$$
$$e_7:(0,0,0,1).$$

Так как векторное поле e_3 коммутирует с полями e_4 и e_7, то из соотношения (2) $[e_3,e_4]=0$ и $[e_3,e_7]=0 \Rightarrow a_3=A_3=\text{const}$, $b_3=B_3=\text{const}$.

Проведя аналогичные преобразования, как для первого типа базиса, получим вырождение.

Рассмотрим третий тип базиса:

$$e_1:(1,0,0,0), e_2:(0,1,0,0), e_3:(0,0,1,0), e_4:(a_4(z_4),b_4(z_4),c_4(z_4),0),$$
$$e_7:(0,0,0,1).$$

Векторное поле e_4 коммутирует с полем e_7, значит, $a_4=A_4=const$, $b_4=B_4=const$, $c_4=C_4=const$.

Аналогично первому или второму типу базиса, получаем вырождение.

И, наконец, четвертый тип базиса:

$$e_1:(1,0,0,0), e_2:(0,1,0,0), e_3:(0,0,1,0), e_4:(a_4(z_4),b_4(z_4),c_4(z_4),0),$$
$$e_7:(a_7(z_4),b_7(z_4),c_7(z_4),0).$$

Векторное поле e_6 коммутирует с полями e_1,e_2 и $e_3 \Rightarrow$ его можно записать в виде: $e_6:(a_6(z_4),b_6(z_4),c_6(z_4),d_6(z_4))$.

Рассмотрим коммутатор $[e_6,e_7]=e_6$:

$$[e_6,e_7]=e_6(e_7)-e_7(e_6)=d_6(z_4)\cdot(a_7'(z_4),b_7'(z_4),c_7'(z_4),0)-0=$$
$$=(a_6(z_4),b_6(z_4),c_6(z_4),d_6(z_4)).$$

Получаем, $d_6(z_4)\cdot 0=d_6(z_4)\Rightarrow d_6(z_4)=0$.

Таким образом, в шести базисных векторных полях последний коэффициент равен нулю. Систему (1) для этой шестерки линейно независимых (над R) базисных полей можно переписать в вещественных координатах. Коэффициенты при дифференцированиях по шести вещественным переменным $x_1=\operatorname{Re}z_1$, $x_2=\operatorname{Re}z_2$, $x_3=\operatorname{Re}z_3$, $y_1=\operatorname{Im}z_1$, $y_2=\operatorname{Im}z_2$, $y_3=\operatorname{Im}z_3$ образуют невырожденную матрицу (6×6). Следовательно, $\partial\Phi/\partial x_1=\partial\Phi/\partial x_2=\partial\Phi/\partial x_3=0$ и $\partial\Phi/\partial y_1=\partial\Phi/\partial y_2=\partial\Phi/\partial y_3=0$ т. е. определяющая функция $\Phi(z_1,z_2,z_3,z_4)$ не зависит ни от одной из трех комплексных переменных z_1,z_2,z_3,z_4, а сама поверхность M вырождена.

Заключение

При исследовании задачи было обнаружено, что, в отличие от начальной ее постановки, заданное семейство 7-мерных алгебр Ли имеет только вырожденные орбиты. Предложенное доказательство этого утверждения является общим для всех алгебр семейства. Оно основано на рассмотрении всех возможных случаев упрощения базисов 5-мерных абелевых подалгебр изучаемых алгебр.

Список литературы

1. Винберг, Э. Б. Курс алгебры / Э. Б. Винберг. – 2-е изд., испр. и доп. – М. : ФАКТОРИАЛ ПРЕСС, 2001. – 544 с.

2. Атанов, А. В. Разложимые пятимерные алгебры Ли в задаче о голоморфной однородности в С3 / А. В. Атанов, А. В. Лобода // Итоги науки и техн. Сер. Соврем. мат. и ее прил. Темат. обз. – 2019. – № 173. – С. 86-115.

Разработка приложения для построения оптимального маршрута в путешествиях

П. Н. Шевлякова

Студент магистрант

П. С. Лысачев

Старший преподаватель

Введение

На протяжении последних десяти лет количество туристов в мире непрерывно растет. Только в России с 2008 года число путешественников увеличилось вдвое и превысило 100 миллионов человек. В настоящее время набирает популярность тенденция путешествовать самостоятельно, а не покупать готовые туры в агентствах.

В интернете можно найти большое количество путеводителей по туристическим местам с подробным описанием маршрута и разнообразными историческими справками. Однако, такие путеводители имеют следующие недостатки – невозможность самостоятельно выбрать достопримечательности и способ перемещения (личный автомобиль, общественный транспорт, пешая прогулка). Во всемирной паутине имеются десятки сервисов для планирования поездок, но многие из них

имеют ограниченную функциональность или неточности в работе. Чаще всего сервис предоставляет строго определенный набор достопримечательностей, без возможности добавления новых мест, либо отсутствует возможность показать, как добраться из одной точки маршрута до другой.

Особое внимание следует обратить на поездки продолжительностью более одного дня, когда выбранные к посещению места необходимо сгруппировать по какому-либо признаку и предложить посетить их в один день. Существующие сервисы либо не предоставляют такой возможности, либо предполагается, что пользователь самостоятельно распределит точки маршрута. В связи с этим существует необходимость создания приложения, позволяющего строить маршрут на заданное количество дней по точкам, определенным пользователем.

1. Цель эксперимента

Цель эксперимента в данной статье – разработка приложения, позволяющего

– Составлять маршруты по выбранным на карте точкам на заданное количество дней;

– Выбирать средство передвижения и в зависимости от этого менять маршрут;

– Получать пошаговое руководство для передвижения на протяжении всего маршрута.

2. Анализ аналогов

Для определения функциональных и нефункциональных требований к приложению необходимо рассмотреть приложения-аналоги. Для анализа были взяты самые популярные сервисы.

– inspirock.com – веб-приложение для планирования поездок. Пользователю необходимо указать город, продолжительность поездки и желаемую «интенсивность». После чего пользователю предоставляется расписание путешествия. В приложении можно также бронировать жилье и покупать билеты. Однако, на сервисе нет возможности выбрать способ передвижения, а сами маршруты не всегда выстроены оптимально.

– evertravel.me – на сайте есть возможность выбрать достопримечательности на карте из списка и распределить их по дням. Порядок посещения выбранных мест не оптимизируется, маршруты для городского или личного транспорта не предоставляются.

3. Анализ и выбор алгоритмов

Для построения оптимального маршрута по заданным точкам на определенное количество дней сначала необходимо распределить точки по дням, то есть выполнить кластеризацию. В ходе анализа были рассмотрены следующие алгоритмы

- Алгоритм выделения связных компонент [1];
- Кратчайший незамкнутый путь;
- K-средних.

Так как в создаваемом приложении число кластеров заранее известно и равняется числу дней, которое заложено на поездку, то подходящим является алгоритм k-средних. Выбранный алгоритм обладает следующими преимуществами – возможность разбиения на кластеры произвольной формы и быстротой работы [2].

После выполнения кластеризации возникает задача обхода точек в оптимальном порядке, начиная с некоторой исходной точки. Под критерием оптимальности будем понимать минимизацию общего пройденного расстояния. Данная задача известна как задача коммивояжера [3]. Рассмотрим возможные алгоритмы решения этой задачи.

- Полный перебор;
- Метод ближайшего соседа;
- Метод ветвей и границ [4].

Точное решение задачи коммивояжера дает только полный перебор, однако время его работы растет экспоненциально с увеличением размера матрицы. Метод ближайшего соседа относится к классу жадных алгоритмов, что зачастую приводит к локальному, а не глобальному минимуму, в связи с чем для реализации был выбран метод ветвей и границ. Подробное сравнение методов дано в [5].

4. Архитектура и средства реализации

Создаваемое приложение строится на клиент-серверной архитектуре. Серверная часть (back-end), где происходят все вычисления, реализована с использованием фреймворка spring-boot. Клиентская часть (front-end), предоставляющая интерфейс для взаимодействия с пользователем, реализована с использованием фреймворка Angular 7. Также были использованы дополнительные библиотеки, такие как

- Google Map API;
- Angular Google Map;
- Ng-bootstrap.

5. Реализация

Для реализации приложения была создана диаграмма классов, изображенная на рис. 1.

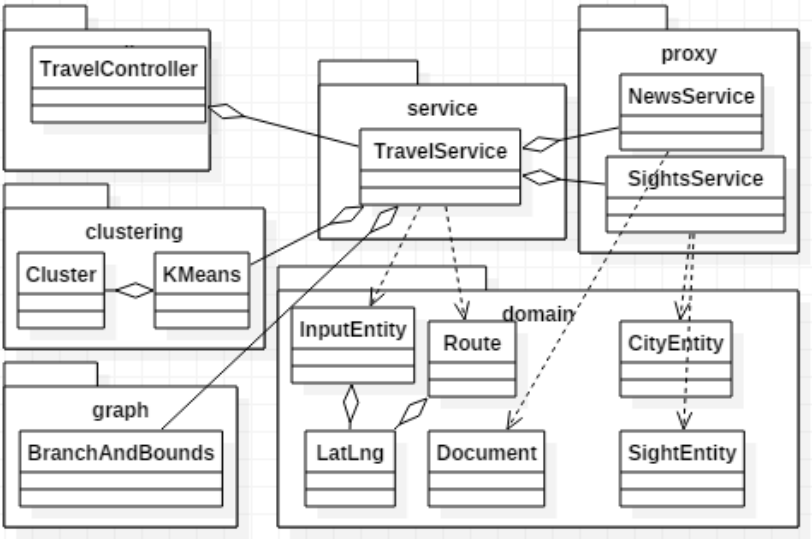


Рис. 1. Диаграмма классов приложения

Основной точкой входа в приложение является класс TravelController из пакета controller. В классе TravelService реализуется основная бизнес-логика приложения. Именно в этом классе осуществляется валидация данных, вызывается метод кластеризации и построения оптимального маршрута.

Пакеты clustering и graph содержат алгоритмы и вспомогательные методы для кластеризации и реализацию алгоритма ветвей и границ соответственно. В пакете domain находятся классы, представляющие собой сущности для передачи данных от клиента серверу. Так класс InputEntity содержит входные данные, полученные от пользователя, класс LatLng – географическая точка с широтой и долготой, класс Route – сущность для представления маршрута. Также имеется пакет proxy со вспомогательными сервисами NewsService для формирования новостной ленты на главной странице приложения и SightsService для поиска достопримечательностей по названию города и формированию подсказок пользователю. Данные два сервиса оперируют сущностями Document, SightsEntity и CityEntity соответственно.

На клиентской части был реализован интерфейс, включающий следующие страницы

– Страница авторизации;

– Главная страница с новостной лентой по тематике «путешествия»;

– Диалоговое окно для создания нового маршрута;

– Страница с сохраненными маршрутами пользователя;

– Страница для отображения подробной информации о сохраненном маршруте с возможностью изменения способа передвижения.

При построении маршрута пользователю необходимо ввести следующие данные:

– Город и продолжительность поездки в днях;

– «Домашнюю» точку – гостиницу или другое место проживания в течение поездки;

– Предполагаемые места для посещения, которые необходимо отметить на карте маркерами, либо выбрать из списка предложенных.

После построения маршрута пользователю будет предоставлен маршрут на каждый день поездки с возможностью выбора средства передвижения.

Экран с построенным маршрутом показан на рис. 2 на примере города Москва. Вверху находится блок для выбора дня, под картой можно выбрать способ передвижения, а также просмотреть подробные инструкции.

Заключение

Данная статья посвящена разработке приложения по составлению оптимального маршрута на заданное количество дней. Описан анализ приложений-аналогов и средств реализации. В результате разработки получилось приложение, позволяющее пользователю самостоятельно планировать поездки на заданный период времени. Данное приложение минимизирует время на планирование поездки и предоставляет всю необходимую информацию о маршруте, что является неоспоримым преимуществом перед аналогами.

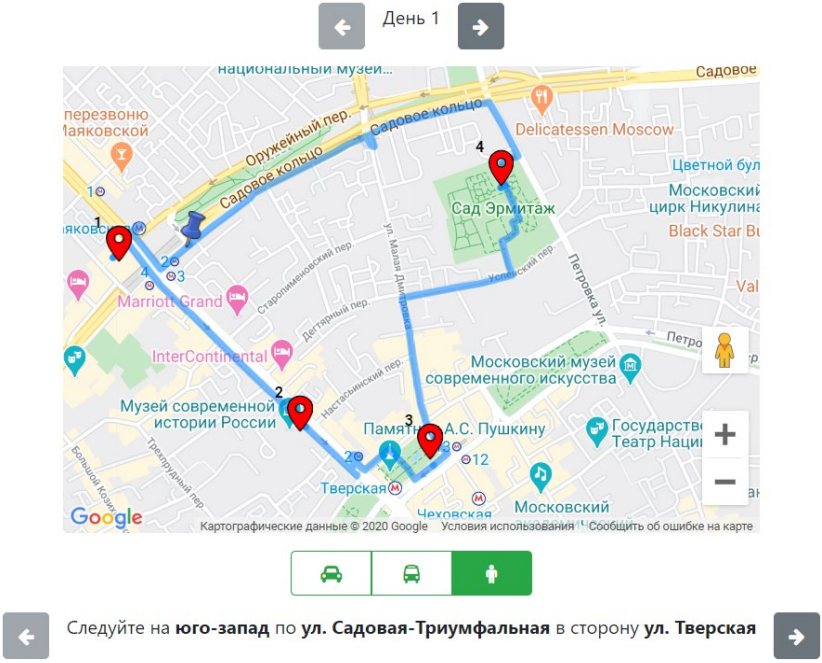


Рис. 2. Экран с построенным маршрутом

Список литературы

1. Лагутин, М. Б. Наглядная математическая статистика : учебное пособие / М. Б. Лагутин. – 3-е изд., испр. – М. : БИНОМ ; Лаборатория знаний, 2013. – 472 с.

2. Мандель, И. Д. Кластерный анализ / И. Д. Мандель. – М. : Финансы и статистика, 1988. – 176 с.

3. Кристофидес, Н. Теория графов. Алгоритмический подход / Н. Кристофидес. – М. : Мир, 1978. – 429 с.

4. Мудров, В. И. Задача о коммивояжере / В. И. Мудров. – М. : Знание, 1969. – 61 с.

5. Гараба, И. В. Сравнительный анализ методов решения задачи коммивояжера для выбора маршрута прокладки кабеля сети кольцевой архитектуры [Электронный ресурс] / И. В. Гараба // Молодежный научно-технический вестник. – М. : ФГБОУ ВПО «МГТУ им. Н. Э. Баумана». – 2013. – №11. – Режим доступа : http://ainsnt.ru/doc/636966.html

Анализ и применение систем билинейных уравнений

М. А. Шмелев

Студент магистрант

А. Ф. Клинских

Профессор

Введение

В данной работе рассматривается задача оптимального плана выпуска продукции, но с учетом брака. Она является хорошо известной задачей линейного программирования. Весомый вклад в эту область внес советский математик и экономист Леонид Витальевич Канторович [1].

Новизна работы состоит в исследовании задачи оптимального выпуска продукции при условии возможного брака. В дальнейшем будет показано, что система как реализация возможных математических моделей является билинейной.

Изложим смысл новой задачи. Перед этим зададим параметры системы: x_i – число продуктов конкретного вида; S_j – число ресурсов конкретного типа, которые тратятся на производство продуктов ($x_1 - x_7$), причем число S_j меньше числа x_i; y_k – параметр, характеризующий наличие брака при изготовлении конкретного продукта.

Если $y_k = 1$, то брака нет. Если $y_k > 1$, то брак присутствует (зависит в последующий момент времени от предыдущего значения).

Если $y_k < 1$, то имеем в наличии более экономичное производство: меньше ресурсов нужно для производства продукта.

Заметим, что y_1 и y_2 относятся к работникам, y_3 и y_4 – к оборудованию, а y_5, y_6 и y_7 – к сырью. Произведение коэффициентов a_{ji} на y_k характеризует число ресурсов, идущих на производство конкретного изделия.

Сформулируем саму задачу. Имеется предприятие по производству изделий для широкого применения. Используются следующие ресурсы: рабочая сила, оборудование и сырье четырех видов: сырье №1, №2, №3, №4 ($S_1 - S_6$). Выпускается семь видов секретных изделий: изделия №1,

№2, …, №7 ($x_1 - x_7$). Цель – получить максимум выпуска двух выбранных изделий ($x_4 * x_7$).

Перед нами модифицированная задача линейного программирования, в которую добавили параметр брака. Благодаря этому она становится билинейной.

Также в нашей статье будет рассмотрено влияние на поведение целевой функции стартовых значений y_i^0, где $t = 0$ характеризует время, i – компонент вектора y. Для этого выделим три сценария поведения системы:

1. Выбор начальных значений на основе оценки из реалий;
2. Расчет на основе равномерного случайного распределения;
3. Расчет на основе гауссова распределения.

1. Метод пошагового решения

Для решения систем нелинейных уравнений удобно использовать базисы Гребнера. Они лежат в основе алгоритма Бухбергера, задействованного в новом методе данной статьи – методе пошагового решения (см. [2, 3]).

Известно, что алгоритм Бухбергера является обобщением метода Гаусса, применяющегося для решения линейных уравнений, и алгоритма Евклида, находящего наибольший общий делитель двух многочленов от одной переменной.

Чтобы найти решение билинейной системы на разных итерациях для задачи оптимального плана выпуска продукции с учетом возможного брака, использовался собственный метод – метод пошагового решения. Условно выделим этапы данного метода:

1. Упрощение системы путем применения метода Гаусса. Вводятся переменные $t_1 \text{K}\, t_{i^2}$:

$$\begin{pmatrix} t_1 & t_2 & \text{K} & t_i \\ t_{i+1} & t_{i+2} & \text{K} & t_{2i} \\ \text{M} & \text{M} & \text{M} & \text{M} \\ t_{i^2-i+1} & \text{L} & \text{L} & t_{i^2} \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 \\ \text{M} \\ x_i \end{pmatrix} \cdot \begin{pmatrix} y_1 & y_2 & \text{K} & y_i \end{pmatrix}.$$

2. Вводим переменные k_j, где $j = 2 \cdot (i-1)$:

$$\begin{pmatrix} k_1 \\ k_2 \\ \text{M} \\ k_{i-1} \end{pmatrix} = \frac{1}{x_i} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \text{M} \\ x_{i-1} \end{pmatrix}; \begin{pmatrix} k_i \\ k_{i+1} \\ \text{M} \\ k_{2\cdot(i-1)} \end{pmatrix} = \frac{1}{y_i} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \text{M} \\ y_{i-1} \end{pmatrix}.$$

Переменные x_i и y_i объявляются «свободными» переменными. Составляется таблица из k_j (каждый t_{i^2} выражается через k_j; в таблице длины строки и столбца равны j), выбираются подходящие строка и столбец. На основе такого выбора часть уравнений, полученных на первом этапе, решается методом Гребнера.

3. Уточнение множества решений путем подстановки в оставшиеся уравнения.

Для нашей билинейной системы достаточно вместо второго этапа применить более простую технику:

1. Подбирается часть уравнений (в нашем случае убирается только одно уравнение из шести) из тех, что были получены на первом этапе;
2. Определяется порядок задания уравнений, а затем переменных в базисе Гребнера (применительно к системе Maple).
3. После решения части уравнений (применяется метод Гребнера) подставляются оставшиеся уравнения (в нашем случае остается только одно уравнение).

2. Исследование билинейной системы

Для наглядности составим одно из уравнений, составляющих систему, без подстановки конкретных значений:

$$S_1 = a_{11} * x_1 * y_1 + a_{12} * x_2 * y_1 + a_{13} * x_3 * y_1 + a_{14} * x_4 * y_2 + a_{15} * x_5 * y_2 + a_{16} * x_6 * y_2 + a_{17} * x_7 * y_2 .$$

Если в системе гипотетически отсутствует брак (все $y_k = 1$), то ее можно рассматривать как линейную.

На рис. 1 показаны уравнения, характеризующие связь между параметрами системы в линейном случае. Ввиду особенностей системы Maple численные значения S_j перенесены в правую сторону с противоположным знаком.

При задействовании параметров y_k уже получается билинейная система, как на рис. 2. В пятом уравнении x_1 и x_2 расписываются как сумма произведений $x_i * y_k$, так как сырье №5 для изделий №1 и №2 доставляется несколькими поставщиками.

Параметр y_i^0 позволяет оценить влияние статистического распределения брака на поведение целевой функции. В дальнейшем будет рассмотрены три сценария поведения системы.

```
#рабочая сила
r1 := 2·x1 + 3·x2 + 4·x3 + 6·x4 + 2·x5 + 6·x6 + 3·x7 - 123;
                                        r1 := 2 x1 + 3 x2 + 4 x3 + 6 x4 + 2 x5 + 6 x6 + 3 x7 − 123
#оборудование
r2 := 3·x1 + 5·x2 + x3 + 2·x4 + 3·x5 + 2·x6 + x7 - 57;
                                        r2 := 3 x1 + 5 x2 + x3 + 2 x4 + 3 x5 + 2 x6 + x7 − 57
#сырье
r3 := 5·x1 + x2 + 2·x3 + x5 - x6 - 8;
                                        r3 := 5 x1 + x2 + 2 x3 + x5 − x6 − 8
r4 := 7·x1 + x2 + 4·x3 + x4 − x5 + x7 − 33;
                                        r4 := 7 x1 + x2 + 4 x3 + x4 − x5 + x7 − 33
r5 := 3·x1 + 2·x2 + x3 + 9·x4 + 3·x5 + 3·x6 + x7 -109;
                                        r5 := 3 x1 + 2 x2 + x3 + 9 x4 + 3 x5 + 3 x6 + x7 − 109
r6 := 7·x1 + 6·x2 + 7·x3 + 2·x4 + 6·x5 + x6 + 5·x7 -106;
                                        r6 := 7 x1 + 6 x2 + 7 x3 + 2 x4 + 6 x5 + x6 + 5 x7 − 106
```

Рис. 1. План выпуска продукции при отсутствии брака

```
r1 := 2·x1·y1 + 3·x2·y1 + 4·x3·y1 + 6·x4·y2 + 2·x5·y2 + 6·x6·y2 + 3·x7·y2 - 123;
      r1 := 2 x1 y1 + 3 x2 y1 + 4 x3 y1 + 6 x4 y2 + 2 x5 y2 + 6 x6 y2 + 3 x7 y2 − 123
r2 := 3·x1·y3 + 5·x2·y3 + x3·y3 + 2·x4·y4 + 3·x5·y4 + 2·x6·y3 + x7·y4 - 57;
      r2 := 3 x1 y3 + 5 x2 y3 + x3 y3 + 2 x4 y4 + 3 x5 y4 + 2 x6 y3 + x7 y4 − 57
r3 := 5·x1·y7 + x2·y7 + 2·x3·y5 + x5·y5 - x6·y6 - 8;
      r3 := 5 x1 y7 + x2 y7 + 2 x3 y5 + x5 y5 − x6 y6 − 8
r4 := 7·x1·y6 + x2·y5 + 4·x3·y5 + x4·y5 − x5·y5 + x7·y6 − 33;
      r4 := 7 x1 y6 + x2 y5 + 4 x3 y5 + x4 y5 − x5 y5 + x7 y6 − 33
r5 := 2·x1·y7 + x1·y5 + x2·y5 + x2·y7 + x3·y5 + 9·x4·y5 + 3·x5·y6 + 3·x6·y6 + x7·y6
      -109;
   r5 := 2 x1 y7 + x1 y5 + x2 y5 + x2 y7 + x3 y5 + 9 x4 y5 + 3 x5 y6 + 3 x6 y6 + x7 y6 − 109
r6 := 7·x1·y5 + 6·x2·y5 + 7·x3·y5 + 2·x4·y5 + 6·x5·y5 + x6·y5 + 5·x7·y5 − 106;
      r6 := 7 x1 y5 + 6 x2 y5 + 7 x3 y5 + 2 x4 y5 + 6 x5 y5 + x6 y5 + 5 x7 y5 − 106
```

Рис. 2. План выпуска продукции при наличии брака

Отметим, что для всех трех сценариев y_k задаются следующим образом:

$$y_1 = y_2 = y_5;$$

$$y_3 = y_4 = y_6 = y_7 = y_1 + 0.15.$$

Для первого сценария $y_1^0 = 1.25$. Данное значение взято на основе оценки из реалий (на предприятии каждый пятый ресурс является браком). Для второго сценария $y_1^0 = 1.05$. Значение характеризует равномерное случайное распределение с диапазоном значений $1-1.1$ и математическим ожиданием 1.05. Последний сценарий относится к гауссову распределению с математическим ожиданием 1.05 и дисперсией 0.01. Поэтому $y_1^0 = 1.08$ (доверительный интервал $= 3\sigma$).

Из-за несоответствия числа выпускаемых продуктов (всего 7) и имеющихся ресурсов (всего 6) система неоднозначна.

Заметим, что функция решения системы будет функцией параметра x_5. Цель можно определить следующим образом: анализ экстремумов функции в зависимости от выбранного параметра (в нашем случае x_5).

3. Построение графиков целевых функций

Наглядно изучить особенности нашей билинейной системы при первом сценарии ($y_1^0 = 1.25$) позволяют графики целевых функций (возьмем 1 и 4 итерации, аналитические продолжения) на рис. 4, 5.

График $x_4 * x_7$ для линейного случая (все $y_k = 1$) показан на рис. 3. Отметим, что он будет неизменным для всех трех сценариев. При отсутствии брака разумно выбрать от 4 до 8 изделий №5 (x_5). Такой выбор соответствует максимально возможным значениям целевой функции.

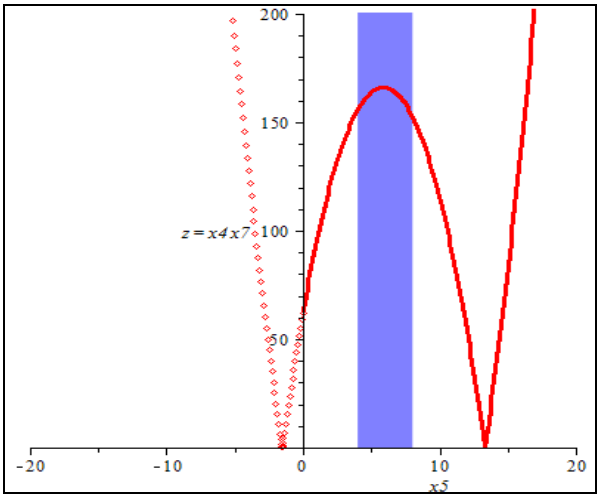


Рис. 3. График целевой функции для линейного случая

При появлении брака на первой итерации (y_1=1.25, y_3=1.4) следует выбрать от 10 до 13 изделий №5. Однако максимальное значение целевой функции (рис. 4) будет в несколько раз меньше максимального значения на предыдущем графике, а, следовательно, число наиболее важных изделий (x_4, x_7) уменьшится. Следующие

итерации также характеризуются малым значением целевой функции. Например, как это показано на рис. 5 (y_1=3.46, y_3=3.61).

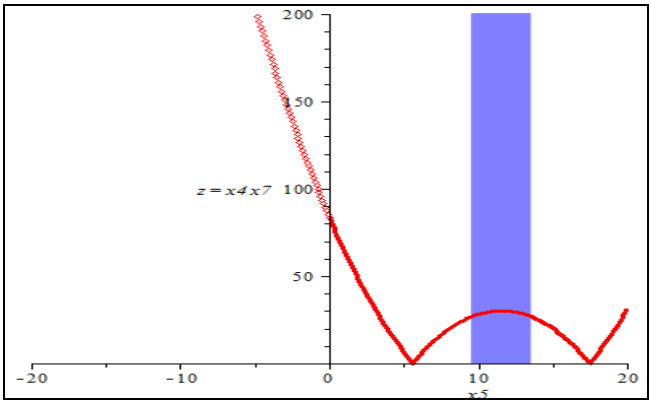


Рис. 4. График целевой функции для первой итерации

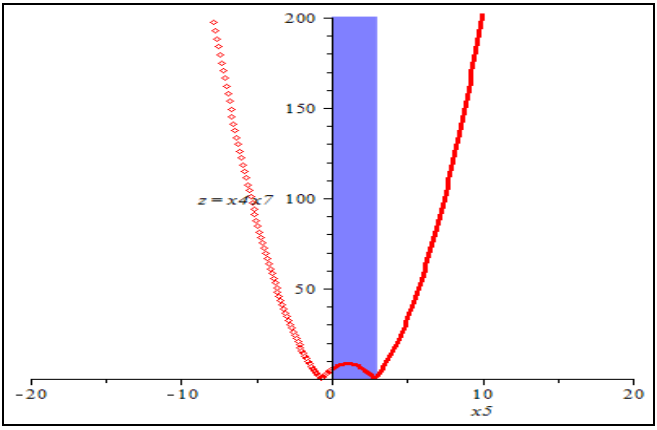


Рис. 5. График целевой функции для четвертой итерации

При анализе особенностей билинейной системы в случае других сценариев следует отметить наличие тех же четырех видов графиков, встречающихся в первом сценарии. Примечательным является тот факт, что при начальном значении, равном 1.08, через две итерации следует повторение итераций из другого сценария при начальном значении 1.25. Это продемонстрировано на рис. 6 (y_1=1.256, y_3=1.406) и рис. 7 (y_1=3.098, y_3=3.248).

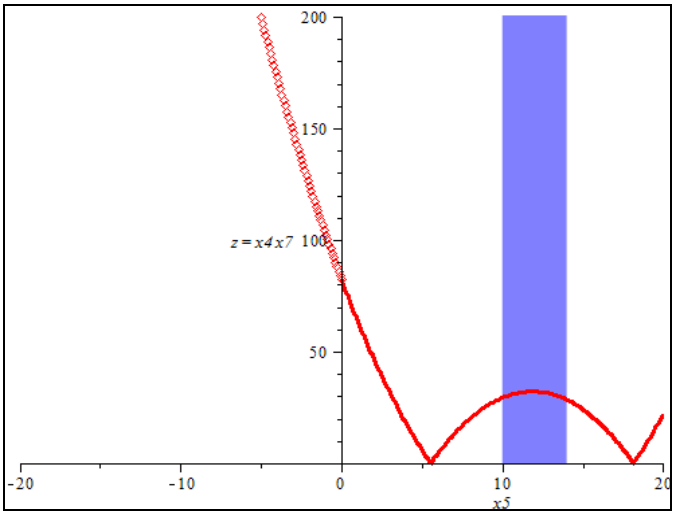


Рис. 6. График целевой функции для третьей итерации и $y_1^0 = 1.08$

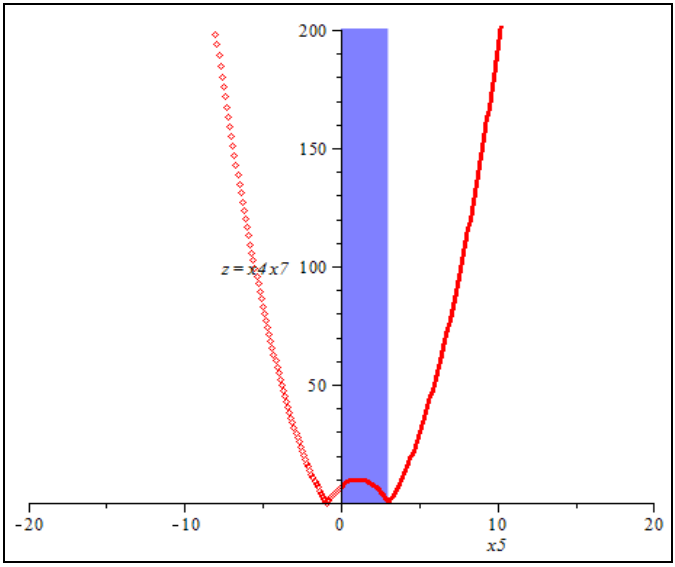


Рис. 7. График целевой функции для шестой итерации и $y_1^0 = 1.08$

Было показано, что величина брака влияет на количество выпускаемых изделий. Чем больше брака, тем продукции меньше. Соответственно, при более экономичном производстве (если $y_k < 1$) можем произвести больше изделий.

Заключение

В рамках данной статьи рассмотрена задача оптимального плана выпуска продукции с учетом возможного брака при разных сценариях.

Изученная билинейная система позволила смоделировать выпуск изделий для предприятия. Учет брака делает систему более гибкой и устойчивой к практическим условиям.

Список литературы

1. Канторович Л. В. Оптимальные решения в экономике / Л. В. Канторович, А. Б. Горстко. – М. : Наука, 1972. – 232 с.

2. Бухбергер Б. Алгоритмический метод в теории полиномиальных идеалов / Б. Бухбергер [и др.] ; под ред. Б. Бухбергера // Компьютерная алгебра. Символьные и алгебраические вычисления. – М. : Мир, 1986. – 392 с.

3. Аржанцев И. В. Базисы Гребнера и системы алгебраических уравнений / И. В. Аржанцев. – М. : МЦНМО, 2003. – 68 с.

Разработка интеллектуального интерфейса, подстраивающегося под критерии пользователя

А. Г. Щербакова

Студент магистрант

Е. Ю. Митрофанова

Доцент

Введение

С момента появления первых систем «человек-машина» и электронно-вычислительной техники возникла проблема взаимодействия человека и компьютера. С развитием информационных технологий усложняются информационные системы, которые применяются во всех сферах деятельности пользователя. Производительность труда оператора напрямую зависит от правильности построения интерфейса информационной системы.

Повышение эффективности человеко-машинных систем требует учитывать все факторы взаимодействия пользователя с компьютером и особенности возникающей при этом среды, где появляются признаки интеллектуального поведения человека и компонентов, генерируемых технической и программной системами [1].

В настоящее время существует потребность в разработке интерфейса, который был бы доступен различным группам пользователей и учитывал их индивидуальные особенности. Существует набор свойств пользовательского интерфейса, наличие которых обеспечивает его способностью самостоятельно адаптироваться к требованиям конкретного пользователя или группы пользователей, а также своевременно подстраиваться под изменения условий функционирования программы, то есть обладать интеллектом [2].

Данная работа посвящена разработке программного средства, реализующего адаптивное поведение интеллектуального интерфейса, позволяющего подстраивать его внешний вид под критерии пользователя с помощью многокритериального метода анализа иерархий.

1. Многокритериальный метод анализа иерархий

К числу наиболее эффективных методов принятия решений в сложных ситуациях относится метод анализа иерархий, предложенный американским ученым Т. Саати. Данный метод опирается на декомпозицию сложной проблемы, на ее более простые составляющие части и дальнейшую математическую обработку последовательности суждений лиц, принимающих решения, которые формируются в виде совокупности парных сравнений [3].

На вершине многокритериальной иерархической структуры находится цель управления, на втором уровне находятся критерии, на третьем уровне расположены подкритерии, самый нижний уровень определяет перечень альтернатив, из которых предстоит сделать выбор.

Пример иерархии, на основе которой в разработанном программном средстве принимается решение о выборе одного из трёх готовых вариантов интерфейса программы, представлен на рис. 1.

После построения иерархии происходит установление приоритетов для элементов всех уровней путём парных сравнений. На следующем этапе формируется набор локальных приоритетов и осуществляется синтез глобальных приоритетов.

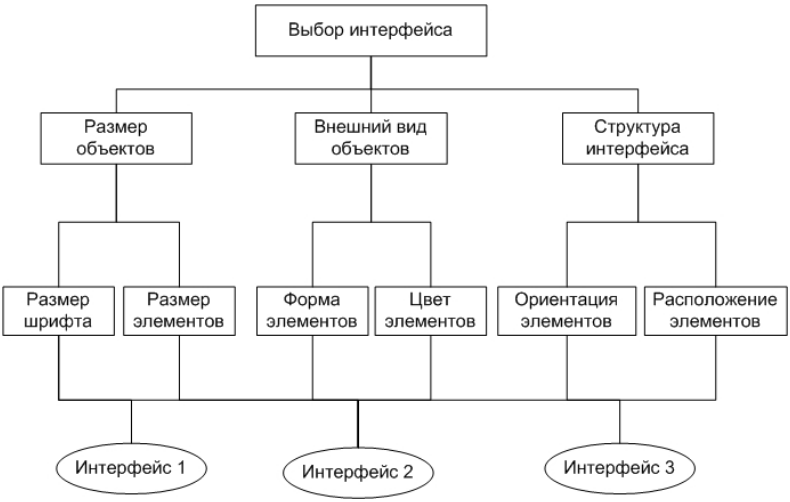


Рис. 1. Формирование иерархии при решении проблемы «Выбор интерфейса»

Определение векторов приоритетов альтернатив относительно критериев второго уровня осуществляется следующим образом:

$$W_{K_1}^A = [W_{ПК_{1.1}}^A, W_{ПК_{1.2}}^A] \times W_{K_1}^E, \quad (1)$$

$$W_{K_2}^A = [W_{ПК_{2.1}}^A, W_{ПК_{2.2}}^A] \times W_{K_2}^E, \quad (2)$$

$$W_{K_3}^A = [W_{ПК_{3.1}}^A, W_{ПК_{3.2}}^A] \times W_{K_3}^E, \quad (3)$$

где $W_{K_1}^E, W_{K_2}^E, W_{K_3}^E$ – вектора локальных приоритетов подкритериев относительно критериев второго уровня; $W_{ПК_{1.1}}^A, W_{ПК_{1.2}}^A, W_{ПК_{2.1}}^A, W_{ПК_{2.2}}^A, W_{ПК_{3.1}}^A, W_{ПК_{3.2}}^A$ – вектора локальных приоритетов альтернатив относительно подкритериев верхнего уровня.

Результирующий вектор приоритетов альтернатив относительно корневой вершины иерархии F вычисляется следующим образом:

$$W_F^A = [W_{K_1}^A, W_{K_2}^A, W_{K_3}^A] \times W_F^E \quad (4)$$

где W_F^E – вектор локальных приоритетов критериев относительно корневой вершины иерархии F.

2. Общий алгоритм настройки интерфейса пользователем

Для решения задачи настройки интерфейса под критерии пользователя необходимо выполнить последовательность действий:

– обработать результаты попарного сравнения критериев пользователем, вычислить вектора локальных приоритетов критериев относительно корневой вершины иерархии;

– обработать результаты попарного сравнения подкритериев для каждого критерия пользователем, вычислить вектора локальных приоритетов подкритериев относительно критериев второго уровня;

– обработать результаты попарного сравнения альтернатив относительно каждого подкритерия, вычислить вектора локальных приоритетов альтернатив относительно подкритериев верхнего уровня;

– вычислить результирующий вектор приоритетов альтернатив с использованием многокритериального метода анализа иерархий.

Набор интерфейсов приложения, которые являются альтернативами в данной задаче принятия решения, проектируется заранее разработчиком. Сервер определяет, какой вариант интерфейса подходит пользователю лучше всего и отображает персонализированную версию сайта. Общий алгоритм работы пользователя с приложением для настройки внешнего вида интерфейса представлен на рис. 2.

3. Реализация программного средства

Разработанное программное средство имеет клиент-серверную архитектуру. В качестве основного средства разработки серверной части приложения был выбран SpringMVC Framework с использованием языка программирования Java. Для реализации алгоритма многокритериального метода анализа иерархий использовалась библиотека математических функций Apache Commons Math с применением пакета org.apache.commons.math3.linear, поддерживающего работу с матрицами и решение систем линейных уравнений. В качестве средств разработки для реализации клиентской части приложения были использованы HTML, CSS и JavaScript.

В соответствии с результатами выполнения алгоритма изменяется дизайн интерфейса веб-приложения путём применения отдельного CSS файла со стилями. Набор CSS файлов предварительно создаётся разработчиком приложения. Файлы определяют внешний вид страницы, размер и расположение объектов, способ представления информации.

На рис. 3 представлена диаграмма вариантов использования, которая иллюстрирует доступные возможности пользователя и эксперта при работе с приложением.

Рис. 2. Общий алгоритм работы пользователя с приложением для настройки внешнего вида графического интерфейса

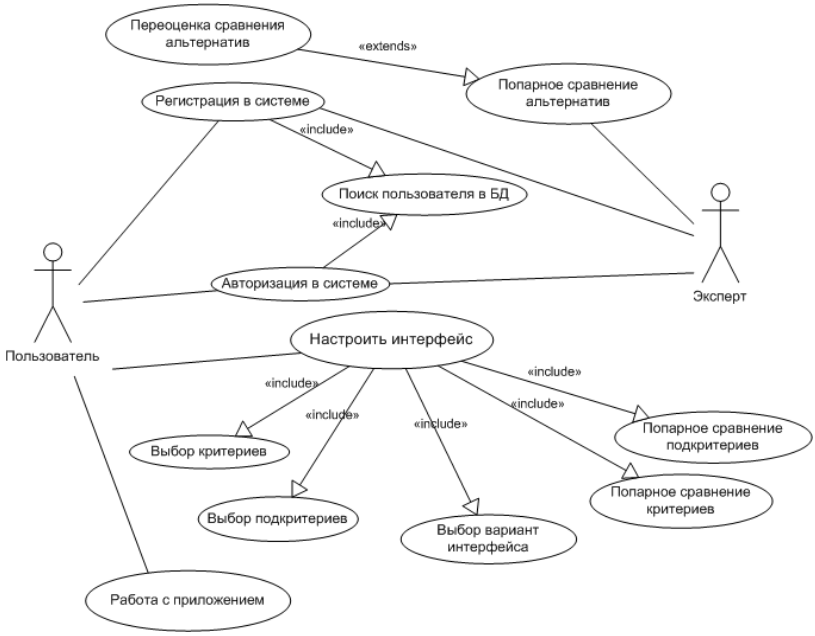


Рис. 3. Диаграмма вариантов использования приложения

Работа с продуктом начинается с момента его запуска на сервере. В случае успешной авторизации пользователя с правами эксперта, открывается страница «Оценка шаблонов интерфейса», изображенная на рис. 4. Здесь в виде таблицы отображается набор критериев и подкритериев, векторы приоритетов и отношения согласованности. Относительно перечисленных подкритериев эксперт может оценить шаблоны интерфейса.

В случае успешного входа в систему пользователя открывается страница, где он выбирает критерии и подкритерии, которые определяют удобство интерфейса. По завершению процесса попарного сравнения критериев и подкритериев происходит переход на страницу «Результат настройки интерфейса», представленной на рис. 5.

Рис. 4. Внешний вид страницы «Оценка шаблонов интерфейса»

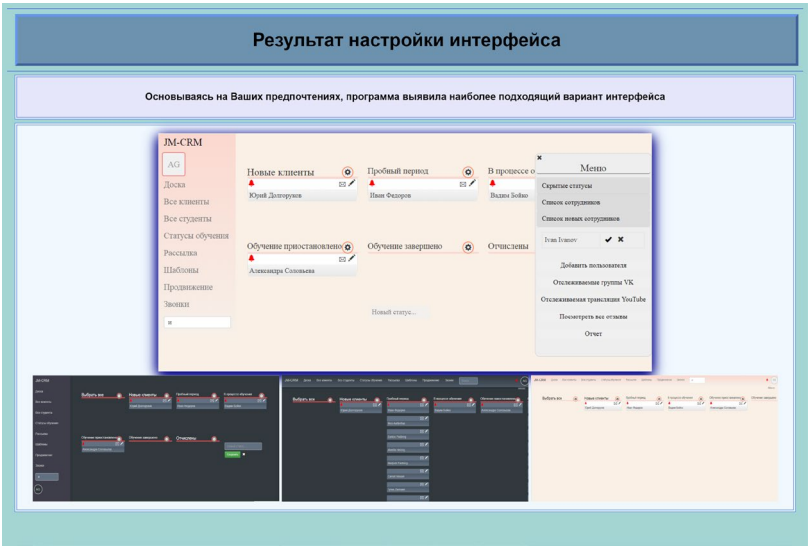


Рис. 5. Внешний вид страницы «Результаты настройки интерфейса»

Система предлагает несколько вариантов интерфейса, расположенных в порядке убывания приоритета, советуя наиболее подходящий с учётом ранее заполненных данных. Пользователь соглашается с системой и выбирает предложенный вариант или не соглашается и выбирает любой другой и продолжает работу с веб-приложением, внешний вид которого теперь соответствует его предпочтениям. Если пользователь уже настраивал интерфейс, то он сразу попадает на главную страницу веб-приложения.

Заключение

Данная статья посвящена разработке и реализации программного средства, реализующего адаптивное поведение интеллектуального интерфейса, позволяющего подстраивать его внешний вид под критерии пользователя. Описан алгоритм принятия решения с помощью многокритериального метода анализа иерархий и создание самого разрабатываемого программного средства. В результате данной разработки получилось рабочее программное средство, реализованное в виде подключаемого модуля. Использование данного решения позволяет настраивать интерфейс программы под индивидуальные предпочтения пользователей. В дальнейшем предполагается использование данного модуля в веб-приложении системы управления взаимоотношениями с клиентами образовательных онлайн-курсов.

Список литературы

1. Кузин, Е. С. Интеллектуальный интерфейс. Общие принципы организации и проблемы реализации / Е. С. Кузин // Известия АН СССР : Техническая кибернетика. – 1985. – № 5. – С. 90-102.

2. Искаков, М. Б. Интеллектуальный интерфейс в информационных системах / М. Б. Искаков // Труды Университета. – 2014. – №4. – С. 74-77.

3. Саати, Т. Принятие решений. Метод анализа иерархий / Т. Саати. – М. : Радио и связь, 1993. – 278 с.

Оценка защищенности автоматизированных систем от деструктивных воздействий

А. В. Юшин

Студент магистрант

В. Ю. Храмов

Профессор

Введение

В статье рассмотрена методика оценки защищенности автоматизированных систем (АС) от деструктивных воздействий и реализующая ее программа для ЭВМ. Данная методика подразумевает использование аппарата нечетких множеств и нечетких продукционных когнитивных карт (НПКК).

Вычислительный процесс на НПКК заключается в выполнении последовательности следующих действий: четкое значение X_i → фаззификация (преобразование к нечеткому виду) → нечеткий логический вывод → получение нечеткого множества для X_j → дефаззификация (вычисление четкого значения X_j), – и т.д. для каждой последующей пары концептов $C_j \rightarrow C_{j+1} \rightarrow \ldots$ на пути следования в НПКК [1].

Для определения взаимного влияния концептов используются нечеткие продукционные правила. Например, переменные X_i и X_j, характеризующие состояния концептов C_i и C_j могут принимать значения из терм-множества T = {Очень_высокая (VH), Высокая (H), Довольно_высокая (MH), Средняя (M), Низкая (L), Очень_низкая (VL)}, задаваемых с помощью соответствующих им функций принадлежности [2, 3].

Основная часть

Основным недостатком НПКК является резкое возрастание числа концептов. Так для определения состояния одного концепта C_k (переменной X_k) при двух концептах C_i и C_j, описываемых соответствующими переменными X_i и X_j, имеем K=2 (количество предшествующих концептов) и M=5 (число термов в множестве T) общее число правил 25.

Однако, введение дополнительного ограничения на функции принадлежности термов позволяет оставлять активными только четыре правила, остальные правила не срабатывают [3].

В качестве показателя защищенности используется вероятность непоражения ЛВС (отдельной ПЭВМ) АС Рнепор., рассчитываемого на основе определения риска информационной безопасности АС [1] при деструктивном воздействии.

$$P_{\text{непор}} = 1 - П_{\text{риск}}, \quad (1)$$

$$П_{\text{риск}} = P_{\text{угр}} * P_{\text{уяз}} * S_{\text{рес}}. \quad (2)$$

$P_{\text{угр.}}$ – вероятность возникновения угрозы, $P_{\text{уяз.}}$ – вероятность наличия уязвимости; $S_{\text{рес.}}$ – ценность (стоимость) информационного ресурса.

Процедура расчета Приск. включает следующие шаги.

Шаг 1. Построение схемы НПКК для рассматриваемого деструктивного воздействия.

Шаг 2. Задание термов переменных состояния в виде лингвистических переменных, принимающих одно из следующих значений: L – Low (Низкая(-ий)); M – Medium (Средняя(-ий)); MH – Medium High (Довольно высокая(-ий)); H – High (Высокая(-ий)); VH – Very High (Очень высокая(-ий)).

Построение функций принадлежности для введенных термов.

Шаг 3. Задание системы нечетких продукционных правил, описывающих состояние концептов C_i в виде матриц риска [1].

П1: Если X_1 есть Низкая и X_2 есть Низкая, то X_4 есть Низкая;

...

П25: Если X_1 есть Очень_высокая и X_2 есть Очень_высокая, то X4 есть Очень_высокая.

Шаг 4. Задание экспертно или экспериментально полученных четких значений входных переменных состояния концептов (на рис. 1 – X1, X2, X3) и построение схемы нечеткого логического вывода.

Допустим, что входные переменные НПКК (т.е. три базовых фактора потенциальной угрозы) принимают значения: $X_1 = 0{,}85$, $X_2 = 0{,}9$, $X_3 = 0{,}75$. Исходя из этих данных определим, что переменные X_1, X_2, X_3, X_4 принимают только значения H и VH, т. е. из 50 правил активными окажутся только 8 правил, соответствующих выделенным блокам из четырех соседних клеток в правом верхнем углу матриц риска.

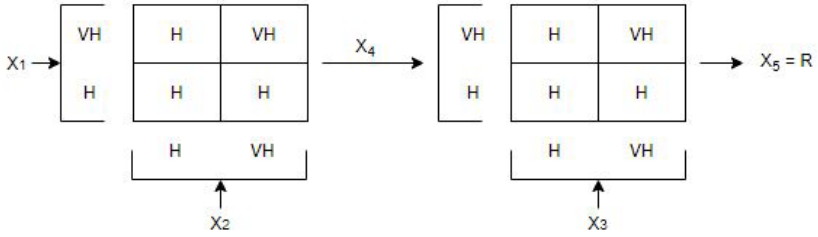


Рис. 1. Пример схемы нечеткого логического вывода

Шаг 5. Нахождение значений промежуточных переменных состояния концептов (на рис. 1 – X4) и переменной риска (на рис. 1 – X5) по алгоритму, представленному на рис. 3.

Шаг 6. Вычисление значения выходной переменной риска по формуле метода взвешенного среднего.

Далее в ходе выполнения работы был разработан алгоритм вычисления промежуточных переменных состояния концептов и переменной риска (алгоритм логического вывода), для реализации которого была разработана программа (рис. 2) с помощью Microsoft Visual Studio на языке программирования C#.

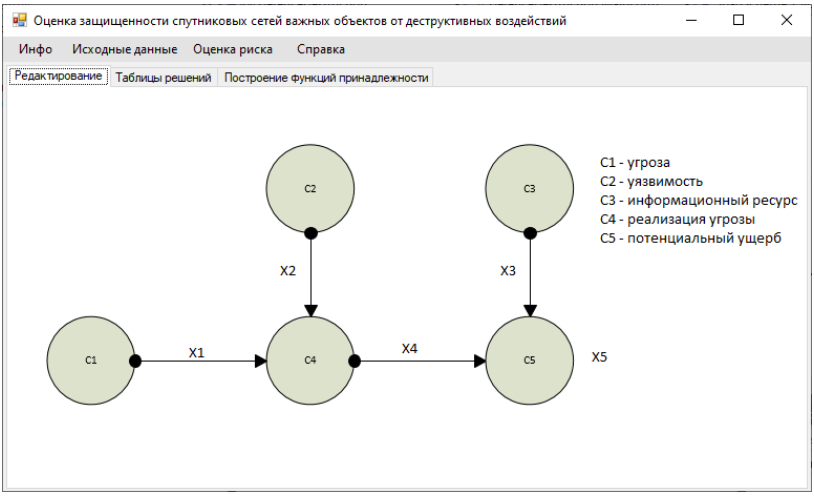


Рис. 2. Редактирование схемы НПКК

Представим схему расчета в виде НПКК, показанную на рисунке выше, где C1 – угроза, C2 – уязвимость, C3 – информационный ресурс,

С4 – реализация угрозы, С5 –потенциальный ущерб, соответственно X_1 – вероятность возникновения угрозы, X_2 – вероятность наличия уязвимости; X_3 – ценность (стоимость) информационного ресурса, X_4 – вероятность успешной реализации угрозы, X_5 – значение ожидаемого потенциального ущерба от СПВ.

Каждая из переменных состояния представляет собой лингвистическую переменную, принимающую одно из следующих значений: L – Low (Низкая (-ий)); M – Medium (Средняя(-ий)); MH – Medium High (Довольно высокая (-ий)); H – High (Высокая (-ий)); VH – Very High (Очень высокая (-ий)). Каждое из этих нечетких подмножеств задается, в свою очередь, собственной функцией принадлежности, построенной с помощью программы, вид которых показан на рис. 3.

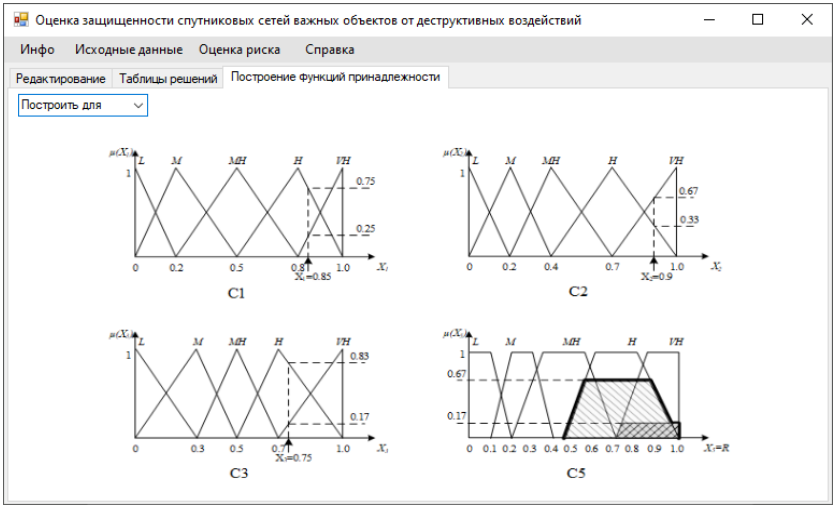


Рис. 3. Построенные функции принадлежности для рассматриваемых нечетких множеств

Используя построенные функции принадлежности, мы можем произвести оценку риска (рис. 4), если предварительно ввести в окна ввода рассчитываемые значения X_1, X_2 и X_3.

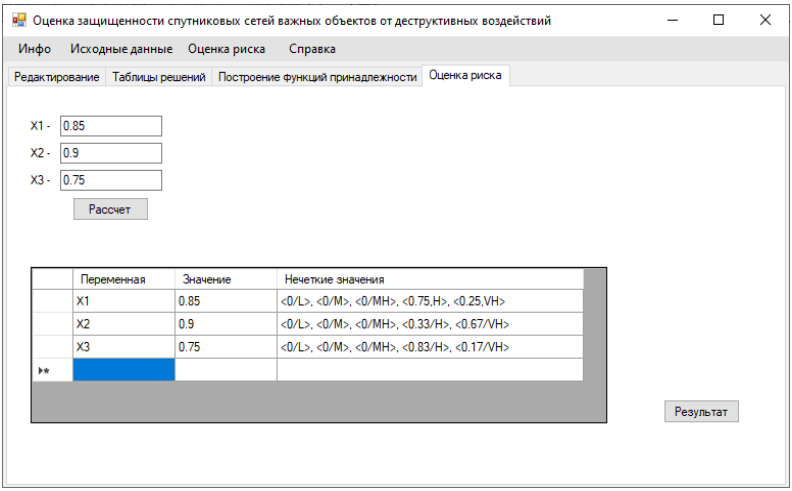


Рис. 4. Производимый расчет нечетких значений для введенных значений

Заключение

В статье рассматривается методика оценки защищенности автоматизированных систем от деструктивных воздействий и реализующая ее программа для ЭВМ, которая была реализована с помощью языка программирования C#. Методика основана на использовании аппарата нечетких множеств и нечетких продукционных когнитивных карт. Она позволяет оценить риск информационной безопасности АС при возможных деструктивных воздействиях, который рассматривается в качестве вероятности защищенности автоматизированной системы.

Список литературы

1. Васильев, В. И. Оценка рисков информационной безопасности с использованием нечетких продукционных карт / В. И. Васильев, А. М. Вульфин, М. Б. Гузаиров // Информационные технологии. – 2018. – Т. 24 – № 4. – С. 266-272.

2. Заде, Л. Понятие лингвистической переменной и его применение к принятию приближенных решений / Л. Заде. – М. : Мир, 1976. – 165 с.

3. Храмов, В. Ю. Косвенные методы построения функций принадлежности системы поддержки принятия решений с нечеткой логикой / В. Ю. Храмов, С. А. Гриценко // Вестник ВГУ. – 2017. – № 4. – С. 126-133.

Содержание

Научное издание

СБОРНИК СТУДЕНЧЕСКИХ НАУЧНЫХ РАБОТ ФАКУЛЬТЕТА КОМПЬЮТЕРНЫХ НАУК ВГУ

Выпуск 14

Часть 2
Научные работы студентов магистратуры

Под редакцией *Д. Н. Борисова*

Издано в авторской редакции

Подписано в печать 16.10.2020. Формат 60×84/16.
Усл. печ. л. 18,8. Тираж 500 экз. Заказ 279

Издательский дом ВГУ
394018 Воронеж, пл. Ленина, 10
Отпечатано с готового оригинал-макета в типографии
Издательского дома ВГУ.
394018 Воронеж, ул. Пушкинская, 3