# Международный журнал информационных технологий и энергоэффективности

Том 7 Номер 2 (24)

2022

Международный журнал информационных технологий и энергоэффективности
Т.7 №2 (24) 2022 г.

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

## СОДЕРЖАНИЕ / CONTENT

## ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ


| | | |
|---|---|---|
| 1. | **Бодренков Г.А., Чернышёв С.А.** Анализ применения мультиагентных систем для задач распределения ресурсов<br><br>**Bodrenkov G. A., Chernyshev S.A.** Application analysis of multiagent systems for resource allocation tasks | **3** |
| 2. | **Мягков А. А., Раскатова М. В.** Исследование эффективности методов классификации для определения мошеннических банковских транзакций<br><br>**Myagkov A.A., Raskatova M.V.** Study of the efficiency of classification methods for identifying fraudious banking transactions | **15** |
| 3. | **Лашков А.А., Зернов М.М.** . Способ и алгоритмы поиска маршрутов протяженных объектов на цифровой карте местности<br><br>**Lashkov A.A., Zernov M.M.** Method and algorithms for finding routes of extended objects on a digital terrain map | **22** |
| 4. | **Тимешов А.С., Раскатова М.В.** Формирование модели генетического алгоритма для решения задачи составления расписания занятий в школе<br><br>**Timeshov A.S., Raskatova M.V.** A model of a genetic algorithm formation for solving the problem of school schedules | **32** |

Бодренков Г.А., Чернышёв .А. Анализ применения мультиагентных систем для задач распределения ресурсов // Международный журнал информационных технологий и энергоэффективности. – 2022. –
Т. 7 № 2(24) с. 3–14

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.94

# АНАЛИЗ ПРИМЕНЕНИЯ МУЛЬТИАГЕНТНЫХ СИСТЕМ ДЛЯ ЗАДАЧ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ

**[1] Бодренков Г. А., [1,2] Чернышёв С. А.**
*[1,2]Санкт-Петербургский государственный экономический университет, Россия (191023, г. Санкт-Петербург, ул. Садовая, 21), e-mail: [1] smart7even@yandex.ru, [2]chernyshev.s.a@bk.ru*
*[2] Санкт-Петербургский государственный университет промышленных технологий и дизайна, Россия (191186, г. Санкт-Петербург, ул. Большая Морская, 18), e-mail: chernyshev.s.a@bk.ru*

**В статье исследовано применение мультиагентных систем для задач распределения ресурсов. Рассмотрены основные аспекты структуры мультиагентных систем и реализации распределения ресурсов в различных сферах с помощью мультиагентных систем, проанализированы их ключевые особенности при выполнении подобных задач и доказана востребованность реализации распределения ресурсов при помощи МАС.**

Ключевые слова: мультиагентные системы, распределение ресурсов, агент, ресурс.

# APPLICATION ANALYSIS OF MULTIAGENT SYSTEMS FOR RESOURCE ALLOCATION TASKS

**[1] Bodrenkov G. A., [1,2] Chernyshev S.A.**
*[1] Saint Petersburg state university of economics, Russian Federation, (191023, Saint-Petersburg, Sadovaya str. 21), e-mail: [1] smart7even@yandex.ru, [2]chernyshev.s.a@bk.ru.*
*[2] Saint Petersburg State University of Industrial Technologies and Design, Russian Federation (191186, Saint-Petersburg, Bolshaya Morskaya str. 18), e-mail: chernyshev.s.a@bk.ru.*

**The article explores usage of multiagent systems for resource allocation tasks. Main aspects of multiagent systems' structure and implementations of resource allocation in different areas using multiagent systems were investigated, key features of multiagent systems in corresponding tasks are analyzed and demand for the implementation of resource allocation using MAS is proved.**

Keywords: multiagent systems, resource allocation, agent, resource.

## Введение

Мультиагентные системы – это системы, состоящие из множества взаимодействующих друг с другом вычислительных элементов, называемых агентами [1]. Такие системы в настоящий момент набирают все большую популярность в различных сферах деятельности, от тяжелого машиностроения до обработки заказов, за счёт сокращения времени работы программы и возможности её упрощения. Рассматривая задачи распределения ресурсов в

различных сферах деятельности, невольно видна параллель между основным принципом мультиагентных систем и поставленной задачей распределения ресурсов, поэтому логично предположить, что такие системы будут подходить для её решения.

**Структура мультиагентных систем.**

Прежде чем посмотреть на применение мультиагентных систем для распределения ресурсов, стоит более подробно разобраться в их структуре. Основной ее элемент – агент. Агентом называют компьютерную систему, находящуюся в некоторой среде и способную к автономной работе в этой среде для достижения поставленных целей [2]. На рисунке 1 представлена схема взаимодействия агента с его средой. Агент берет сенсорные данные от среды, а затем действует в зависимости от них. Это взаимодействие обычно довольно длительное и непрекращающееся. В большинстве случаев агент не имеет контроля над средой и в лучшем случае может частично ею управлять, влияя на неё своими действиями.

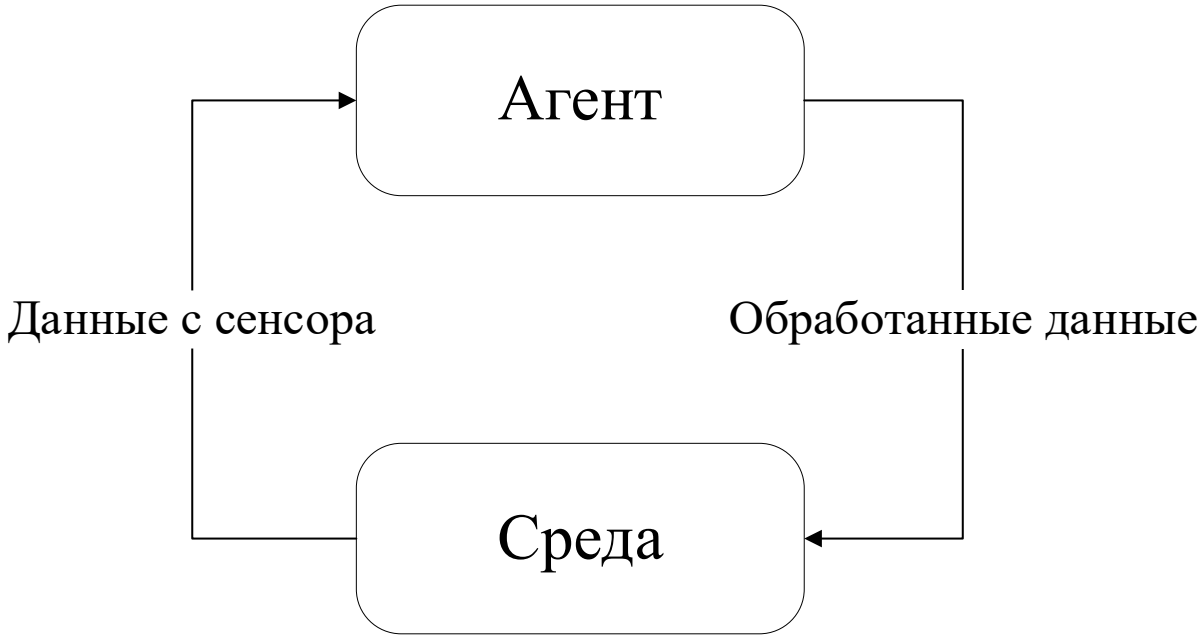


Рисунок 1 – Агент в своей среде

Говоря о разработке систем с хотя бы одним агентом, основной проблемой является его ход действий – выбор того, что ему нужно сделать в зависимости от данных из среды. Существуют две агентных архитектуры:

- операционная, где агенты и мультиагентные системы – системы со своей конкретной характеристиками, структурой и поведением;
- базирующаяся на системных уровнях агенты и мультиагентные системы являются элементами находятся на отдельных уровнях [3].

Этим архитектурам соответствуют различные методологии их построения, предлагающие реализацию рационального принятия решений, такие как MAS-CommonKADS, MaSE, Adelfe, Zeus, INGENIAS и т. д. [4]. Данные архитектуры предполагают то, что агент является «разумным», то есть обладает следующими свойствами [1]:

- реактивность – способность воспринимать свою среду и своевременно отвечать на её изменения в соответствие с заданной целью;

- инициативность – способность к достижению заданной цели путем проявления инициативы;
- социальность – способность к взаимодействию с другими агентами и человеком для достижения заданной цели.

**Общая математическая модель реализации задачи распределения ресурсов мультиагентной системой**

Разобрав структуру и работу мультиагентных систем, можно перейти к общему принципу реализации задачи распределения ресурсов с их помощью. Для этого необходимо постановка конкретной задачи – задачи о распределении ресурсов между заказами для получения максимальной прибыли. По [5], в общем виде её можно представить следующим образом. Пусть есть n заказов и m типов ресурсов. Обозначим $r_j$ — объем j-го ресурса (j = 1, 2, …, m). Будем считать, что для каждого заказа i (i = 1, 2, …, m), задана функция $c_i: R_+^m \to R$, задающая возможную прибыль при использовании $x_{i,1}, x_{i,2}, \dots, x_{i,m}$ ресурсов соответствующего типа. Тогда задача распределения ресурсов сводится к нахождению матрицы ресурсов $X \in R^{nm}$, которая максимизирует функционал

$$C(X) = \sum_{i=1}^{n} c_i \left(x_{i,1}, x_{i,2}, \dots, x_{i,m}\right) \to max$$

при условии выполнения ограничений на общее количество ресурсов

$$\sum_{i=1}^{n} x_{i,j} \leq r_j,\ j = 1, 2, \dots, m,$$

и их неотрицательности

$$x_{i,j} \geq 0, i = 1, 2, \dots, n, j = 1, 2, \dots, m.$$

Для решения этой задачи при помощи мультиагентных системах надо учитывать их децентрализованность: не каждый агент такой системы может взаимодействовать с другими. Следовательно, не каждому заказу может быть доступен каждый из ресурсов, что по [5] задается следующим ограничением: пусть $D \in M(n, m)$ — матрица смежности графа связей, $D_{ij} \in \{0, 1\}$, тогда к ограничениям добавляется

$$x_{i,j} \leq D_{ij} r_j$$

и решением задачи будет являться нахождение максимума функционала

$C(X) \to max, C: R^n \to R.$

Далее будут рассмотрены примеры реализации решения задачи распределения ресурсов в различных сферах деятельности.

**Распределение производственных ресурсов на предприятии тяжелого машиностроения с помощью МАС «Оптимизатор».**

Современное предприятие тяжелого машиностроения является сложным в управлении, ибо необходимо обеспечить требуемого уровня качества как выпускаемой продукции, так и производства в целом, постоянно совершенствуя производство различными средствами. Решение такой задачи может быть мультиагентными.

Для начала следует выделить особенности задачи планирования такого производства [6]:

- сложная структура графиков (множество этапов связаны между собой и должны/могут выполняться параллельно, а также возможное наличие нескольких начал и концов);
- наличие особенного графика работы для каждого ресурса;
- у каждого этапа есть предпочтительный ресурс, в качестве которого используется конкретный рабочий центр;
- при планировании необходимо учитывать загрузку каждого ресурса;
- некоторые этапы не требуют ресурсов (загрузка равна нулю);
- перераспределение этапов может быть как разрешено, так и запрещено.

Такие особенности способна учесть мультиагентная система «Оптимизатор». Эта МАС работает по следующему принципу: всем рабочим центрам (ресурсам) и графикам/этапам (заказам) ставятся в соответствие агенты, действующие в интересах этих заказов и ресурсов. В ходе взаимодействия агентов с ресурсами и другими агентами строится сбалансированный по многим критериям план производства с учетом всех ограничений и предпочтений.

Архитектура данной системы состоит из следующих компонентов:

- модуль Engine, в котором находятся агенты Agents, их свормы (группы, в которых они работают) Swarms, активности (логика и взаимодействия агентов) Activities, события (конкретные действия агентов) Events и средства коммуникации Messaging;
- модуль Domain, который отвечает за связь с другими элементами информационного пространства производства (экспорт результатов работы, загрузка данных в систему и т. д.);
- модуль Scheduler, который реализует логику планирования (изменение времени выполнения этапов, добавление новых, проактивность агентов (возможность принимать решения по планированию самостоятельно, например, передвинуться на более раннее время или ближе к завершению процесса) автоматическое тестирование плана и т. д.);
- модуль Integration, который физически реализует планирование (веб-сервисы, визуальные компоненты, диаграмма Гантта и т. д.);
- модуль визуализации плана в виде линейного графика единого процесса.

Алгоритм выполнения задачи планирования может быть следующим:

- при поступлении заказа создается событие на его планирование и заказ передается агенту;
- агент, получив описание этапов выполнения заказа, создает новых агентов для каждого из этапов;
- агент заказа отправляет последнему агенту этапа сообщение о начале планирования;
- агент этапа, получив сообщение, отправляет агенту соответствующего ресурса запрос на использование ресурсов;
- агент ресурса проверяет свои предпочтения и возможности, и действует в соответствие с ними: если они позволяют, то агент ресурса отправляет сообщение агенту этапа о том, что ресурсами можно воспользоваться, а если не позволяют, то отправляет сообщение с контрпредложением;
- в случае успеха агента этапа сообщает агенту заказа о успешном планировании;
- агент заказа определяет следующий этап по ранее сформированному плану и алгоритм повторяется;

- после формирования всех этапов их агенты могут проактивно улучшить свое состояние по своим предпочтениям.

**Распределение трудовых ресурсов в проекте**

Современные системы управления трудовыми ресурсами поддерживают функцию распределения трудовых ресурсов с позиции их загруженности, но выбор, кому из работников дать определенное поручение в большинстве случаях стоит за менеджером проекта, что порой бывает ненадежно, особенно если менеджер неопытен. Поэтому нужно улучшить систему управления трудовыми ресурсами с учетом последовательности выполнения работы, характеристиками работника и его нагрузкой при помощи внедрения мультиагентных систем [7].

Суть работы мультиагентной системы заключаться в следующем: все задачи и сотрудники представляются отдельными агентами, а для обеспечения их взаимодействия определяются целевые функции, последовательность переговоров и тип взаимодействий. Также мультиагентное решение включает в себя:

- метод распределения трудовых ресурсов, который разрешает распределять ресурсы соответственно сетевому графику и одновременно учитывать пси-характеристики труда работника, такие как быстродействие, уровень знаний, ответственность, надежность и его нагрузку, за счет чего повышается оптимальность распределения трудовых ресурсов и точность оценивания эффективности работы работника;
- математическую модель взаимодействия агентов системы, которая разрешает прогнозировать следствия действия на общее распределение ресурсов;
- математическую модель системы управления и контроля за распределением трудовых ресурсов, которая разрешает учесть особенности трудовых ресурсов и дает возможность автоматизировать процесс контроля и распределения трудовых ресурсов.

С описанием математической модели данной МАС можно ознакомиться в [7].

**Smart Grid (умные сети электроснабжения).**

С растущей интеграцией распределенных энергетических ресурсов в энергосистему, децентрализованный подход к её реализации становится необходим для планирования и распределения ресурсов в умных сетях электроснабжения Smart Grid [8]. К таким ресурсам можно отнести маломасштабные ресурсы спроса или предложения электроэнергии, взаимосвязанные с электрической сетью и обычно расположенные рядом с центрами нагрузки системы.

Экономичное распределение нагрузки и обязательства агрегата являются двумя основными проблемами распределения ресурсов, которые играют решающую роль в безопасной и стабильной работе данной системы. Неопределенность, связанная с возобновляемыми источниками энергии, еще больше усложнила задачу распределения ресурсов для операторов сети. Сеть будущего будет иметь инновационное сочетание возобновляемых источников энергии и бóльшую нагрузку на электрические транспортные средства с возможностью двунаправленного потока энергии. Эта сложная интеллектуальная сетевая система требует разработки децентрализованного подхода к проблеме распределения ресурсов, который позволяет осуществлять межузловую связь и соответствующее принятие

решений, что позволяют реализовать распределенные мультиагентные системы.

В целом, реализация MAC в системе умной сети электроснабжения Smart Grid представляет собой интеграцию физической сети с коммуникационным уровнем, где агенты действуют как интерфейс. Коммуникационный уровень представляет собой сильно связанную сеть с различными настраиваемыми топологиями. Агенты делятся на 3 блока:

- блок устройства;
- блок принятия решений;
- блок связи.

Блок устройства можно рассматривать как физическую шину энергосистемы с такими компонентами, как синхронные генераторы, возобновляемые генераторы, гибкая нагрузка и жесткая нагрузка. Блок принятия решений выполняет локальные вычисления для агентов, а блок связи является коммуникационным узлом, который передает и получает информацию. Внутренняя структура агента состоит из трех блоков:

- коммуникационный блок;
- блок принятия решений;
- блок устройства.

Коммуникационный блок является приемником/передатчиком сигналов, используемых для обмена информацией с соседями. Калькулятор, датчики и контроллеры являются частью блока принятия решений, который отвечает за локальные вычисления в агенте. Блок принятия решений является мозгом узла агента и способен давать инструкции для блока устройства, а также он отвечает за передачу информации в блок коммуникаций. Блок устройства представляет собой традиционную шину в сети, состоящую из различных элементов, таких как синхронные генераторы, возобновляемые генераторы, аккумуляторные накопительные системы, гибкие и жесткие нагрузки. Блок устройства выполняет указания блока принятия решений, а также отправляет ему обратную связь.

Существует множество инструментов моделирования и инструментов с открытым ресурсным кодом для моделирования таких мультиагентных систем. Например, ZEUS, AgentBuilder, JADE, MADKit и т. д.

**Вычислительные центры коллективного пользования**

Широкое распространение в решении прикладных задач приобрели сегодня высокопроизводительные вычислительные центры коллективного пользования (ВЦКП) – объединенные вычислительные среды, предназначенные для обслуживания ресурсных запросов пользователей, состоящие из разнородных вычислительных систем (ВС), администрируемых независимо друг от друга и предоставляющих неотчуждаемые ресурсы для общего пользования [9]. К ним предъявляется множество требований по обеспечению качества обслуживания – загруженности, гарантированному времени выполнения поступающих ресурсных запросов, отказоустойчивости и эффективности работы. Учитывая тот факт, что ресурсы ВЦКП автономны (они обслуживаются владельцами, имеющими право реализовывать независимый доступ к ним) и используются в коллективном режиме, а состав ресурсов, пользователей и их заданий динамично меняется, встает вопрос о реализации распределенной системы управления потоком заданий для ВЦКП, чем и являются мультиагентные системы.

Модель ЦКП включает в себя программные агенты, реализующие модели внешних

источников задач, распределителя заданий, сбора и анализа статистики, вычислительных систем.

Модель внешней среды источников заданий представлена однотипными программными агентами, имитирующими пользователей, отправляющих задания на ЦКП. Для использования модели в системе управления реального ЦКП поток заданий меняется с моделируемого на реальный.

Распределитель заданий представлен агентом коммутатором ЦКП, который выполняет распределение поступивших заданий на отдельные вычислительные системы (кластеры) ЦКП. Агент коммутатор ЦКП периодически получает сообщения от кластеров о состоянии их очереди заданий. При поступлении пользовательского задания агент коммутатор отправляет агенту сбора и анализа статистики информацию о характеристиках задания и пользователе. В ответ он получает средние значения зафиксированных характеристик решенных заданий для данного пользователя и имена ВС, на которых они чаще решались – основная и альтернативная. Оценивая информацию об этих ВС, коммутатор передает задания наиболее подходящей.

Модель отдельных кластеров ЦКП состоит из агентов следующих типов:

- агента коллектора ВС;
- агента контроллера работающего домена ВС;
- агента контроллера домена свободных вычислительных узлов (ВУ);
- агента контроллера резервного домена;
- агента ВУ.

Агент ВУ постоянно собирает информацию о нагрузке ВУ и состоянии линий связи, ведет постоянное наблюдение за ходом выполнения задания на ВУ, регистрируя интенсивность обмена между ветвями параллельной программы, частоту появления тех или иных событий, изменения переменных и т. п. На основании этой информации ведется контроль тенденции изменения параметров ВУ и хода решения задачи и принимается решение о необходимости перераспределения нагрузки или освобождения данного узла.

Получая задание, агент контроллер работающего домена принимает список ВУ для счета и ссылки, помеченные как перспективные. В процессе работы этот агент ведет контроль за исполнением задания. При необходимости перераспределения нагрузки или увеличения ресурсов целевой узел отыскивается из числа ресурсов данной области. Если таковых нет, то взаимодействуя с контроллерами доменов, в которых находятся помеченные как перспективные ВУ, домена свободных ВУ и резервного домена, узнает адрес узла, способного принять дополнительную нагрузку.

Агент контроллер домена свободных ВУ получает сведения от всех ВУ домена об их состоянии. При получении задания производит поиск по иерархическому дереву коммуникационных связей нужного количество ВУ. Также производится поиск ВУ, которые подходят для решения задания, но пока не могут принять участие в вычислениях. После чего передает контроллеру работающего домена полученный список ВУ и само задание. При необходимости выделяет дополнительные ВУ по запросам контроллеров доменов.

Агент контроллер резервного домена содержит адреса вычислительных узлов, отправленных в резерв. Также в этом агенте по запросам от агентов-контроллеров работающих доменов происходит поиск подходящих ВУ, способных удовлетворить их ресурсные требования.

Агент коллектор ВС принимает информацию от контроллеров доменов о получении задания и о завершении выполнения задания. При получении задания агент коллектор просматривает реализующуюся в данный момент очередь заданий. Если не удается включить задание в эту очередь, оно отправляется ожидать формирования новой очереди. При получении сигнала о завершении задания с помощью все того же алгоритма оценивается возможность занять освободившиеся ресурсы заданием из текущей очереди или из ожидающих заданий. Когда текущая очередь подходит к концу, начинается формирование следующей очереди с помощью генетического алгоритма.

**Генезис Знаний**

Генезис Знаний – группа компаний, занимающихся разработкой мультиагентных решений задач распределения ресурсов на предприятиях в реальном времени [10]. Одним из таких программных продуктов является Smart Supply Networks – интеллектуальная система управления сетями поставок.

Главной целью данной системы является решение задачи планирования работы системы снабжения [11]. Она состоит в сбалансированном повышении заданных показателей качества построенного плана в условиях постоянных быстрых изменений. В результате планирования должны быть получены:

- перечень операций по перемещению продукции (количество, продукт, время отправки и прибытия, узел отправки и прибытия), необходимых для выполнения заданных операций;
- перечень операций по производству продукции (количество потребляемого продукта, количество производимого продукта, время начала и окончания, узел сети), необходимых для выполнения заданных операций;
- график потребления продукции по заказам или по прогнозу спроса с учетом допустимых отклонений по времени и наличия продукции по плану снабжения.

Данная МАС реализует следующие основные блоки функций (рисунки 1, 2):

- управление контекстами данных (сетями и их версиями), пользователями и правами доступа пользователей к управляемым сетям;
- интерактивное редактирование структуры сети снабжения – узлов (поставщиков, производственных центров, мест хранения и распределения, мест реализации продукции), каналов поставки между узлами и их параметров (длительности и себестоимости поставки по каналу, емкость складов, стоимость хранения);
- редактирование перечня продукции и технологических процессов производства (требуемых материалов, промежуточных результатов, готовой продукции, отходов производства, длительности и себестоимости производственных операций, требуемых производственных линий);
- редактирование текущих остатков продукции в сети на различных узлах;
- редактирование фиксированных и утвержденных производственных операций;
- редактирование фиксированных и утвержденных транспортных операций;
- редактирование заказов на поставку продукции или прогноза спроса на продукцию по узлам сети;
- динамическое адаптивное планирование снабжения (обеспечения заказов, спроса и выполнения фиксированных операций в сети) путем построения оптимизированных

по показателям эффективности планов закупки, хранения, производства и перемещения продукции в сети с учетом заданных ограничений;

- просмотр и анализ результатов планирования, достигнутых показателей эффективности, отклонений полученного плана от предпочтений, микроэкономики выполнения заказов;
- ручная интерактивная корректировка построенных системой планов (фиксация отдельных частей плана, ввод управляющих ограничений на производство и перемещения по отдельным каналам и участкам сети) с автоматической адаптацией плана под внесенные изменения;
- отслеживание выполнения построенных планов, ввод данных по фактическому началу и окончанию операций, фиксация отклонений от плана и перепланирование в реальном времени с учетом отклонений и минимизацией потерь эффективности.
- Архитектура данной МАС состоит из трех слоев:
- слой данных, обеспечивающий постоянное надежное хранение данных, не зависящее от функционирования остальных частей системы;
- средний слой, включающий подсистемы, обеспечивающие выполнение бизнес-логики в процессе использования системы, модуль планирования, функции интеграции с внешними системами, разделение прав доступа, другие серверные функции;
- слой визуализации и взаимодействия с пользователями, включающий подсистемы, обеспечивающие интерактивную работу системы с пользователями на различных устройствах, ввод, редактирование и визуализацию данных, результатов планирования и анализа.

В основе взаимодействия агентов лежит принцип локального построения планов. Перепланирование (как, в целом, и начало планирования) начинается с получения события агентом узла и передачи управления агентами потребностей, которые затронуты поступившим событием. Их основная цель – найти наиболее выгодный вариант планирования при сложившихся условиях. Агенты потребностей пытаются удовлетвориться за счет возможностей, доступных на их узле, взаимодействуя с агентами хранилища, сформированными перевозками, и агентами входящих каналов, формирующими входящие перевозки. После построения локального варианта плана управление передается агенту узла, который координирует подпотребности, возникшие в результате локального планирования, с агентами других узлов, которые должны отправить продукцию по каналу. Затронутые узлы формируют своих агентов подпотребностей и тоже запускают локальное планирование.

**Яндекс Маршрутизация**

Яндекс Маршрутизация – сервис, основанный на мультиагентных системах, позволяющий решать логистические задачи распределения ресурсов [12]. Он помогает распределять заказы и составлять маршруты перевозки с учётом различных факторов, таких как пробки, интервалы доставки, весогабаритные характеристики автомобилей и т. д. В нее входит два продукта: автоматическое планирование доставки и мониторинг выполнения заказов. Планирование доставки работает следующим образом [13]:

- логист передает программе данные о машинах/курьерах, заказах и складе в Excel или в запросе к программному интерфейсу приложения (API);

- алгоритм формирует стоимость решения из стоимости используемых транспортных средств, которая высчитывается двумя способами:
    1) по типовой формуле, которую предоставляет сервис. Для большинства ситуаций эта формула отражает структуру реальных затрат на маршрут: есть какая-то фиксированная стоимость использования машины (стоимость ее использования и выполнения одного рейса) и есть переменная составляющая, которая зависит от пробега/времени/количества заказов/веса груза. На практике чаще всего используются фиксированная стоимость использования машины, стоимость за километр пробега и стоимость за час работы, они и содержат определенные значения по умолчанию. Стоимость за тонно-километр транспортной работы используется для сокращения пробега с тяжелыми грузами. Он учитывается при расчете суммарной стоимости транспортной работы;
    2) по формуле, которая задана для конкретного транспортного средства пользователем программного продукта;
- алгоритм формирует стоимость штрафов. Они могут быть различны, от штрафа за недоставку заказа до недостаточной сгрупированности маршрута. Значения штрафов по умолчанию позволяют получить хорошие результаты, но для наилучших результатов стоит поменять их под конкретную ситуацию;
- программа минимизирует значение, сложенное из стоимости решения и стоимости штрафов.
- Данный алгоритм отличается некоторыми особенностями, такими как:
- вероятностный подход к решению задачи. На практике это означает, что на одном и том же наборе исходных данных результат планирования может быть различным, но итоговое значение стоимости решения будет отличаться от других возможных незначительно;
- наличие жестких и мягких ограничений;
- предсказуемое время работы алгоритма.

**Вывод**

В данной статье рассмотрено применение мультиагентных систем в решении задач распределения ресурсов. МАС продолжают показывать свою эффективность в данной сфере и в смежных с ней задачах вроде мониторинга работы и планирования дальнейших действий. К тому же с учетом постоянного возрастания сложности производства на предприятиях, мультиагентные системы будут становиться только более востребованными, что делает их наиболее подходящим решением задачи как распределения ресурсов, так и смежных с ней сфер.

**Список литературы**

1. M. Wooldridge. An Introduction to Multiagent Systems. – Wiley Publishing, 2009. – pp. 19-28.
2. G. Weiss. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. – The MIT Press, 2000. – pp. 27-30.
3. M. Oprea. Applications of Multi-Agent Systems // Conference paper of the IFIP International Federation for Information Processing book series (IFIPAICT, volume 157), Boston, 2004. –

pp. 244-246.
4. Jorge J. Gómez-Sanz, Juan Pavón. Methodologies for Developing Multi-Agent Systems // Journal of Universal Computer Science, Graz University of Technology, Austria, 2004. – pp. 370-371.
5. Н. В. Мальковский, О. Н. Граничин, К. С. Амелин. Распределение ресурсов в контексте мультиагентных систем // XII Всероссийское совещание по проблемам управления, Москва, 2014. – с. 9004-9010.
6. М. В. Андреев, А. В. Иващенко, П. О. Скобелев. Мультиагентная система распределения производственных ресурсов в тяжелом машиностроении // Журнал «Программные продукты и системы», 2010. – с. 101-102.
7. Е. А. Зинец. Метод и средства создания мультиагентной системы управления и контроля за распределением трудовых ресурсов // Журнал «Научные труды Винницкого национального технического университета», 2009. – с. 1-6.
8. A. S. Nair T. H. Mitch C. D. Flora S. N. Goveas N. Kaabouch P. Ranganathan Multi-Agent Systems for Resource Allocation and Scheduling in a Smart Grid // Journal «Technology and Economics of Smart Grids and Sustainable Energy», Singapore, 2018. – pp. 2-4.
9. Д. В. Винс. Мультиагентная модель системы оперативного управления распределением ресурсов для ВЦКП // Труды конф. молодых ученых. Новосибирск, 2013. – с. 41-44.
10. Генезис Знаний Цели и задачи [Электронный ресурс]. URL: http://www.kg.ru/company/about/ (дата обращения 5.05.2022 г)
11. А. В. Царев. Развитие МАС Smart Supply Networks для учета особенностей производственно-транспортной логистики и возможности управления крупными сетями снабжения с обработкой данных большой размерности // Труды XVIII Международной конференции «Проблемы управления и моделирования в сложных системах», Самара, 2016. – с. 223-236.
12. Яндекс.Маршрутизация: сервис автоматического планирования маршрутов [Электронный ресурс]. URL : https://yandex.ru/routing/vrp (дата обращения 5.05.2022 г)
13. Начало работы с Яндекс.Маршрутизацией [Электронный ресурс]. URL: https://yandex.ru/routing/doc/vrp/ (дата обращения 5.05.2022 г)

**References**

1. M. Wooldridge. An Introduction to Multiagent Systems. – Wiley Publishing, 2009. – pp. 19-28.
2. G. Weiss. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. – The MIT Press, 2000. – pp. 27-30.
3. M. Oprea. Applications of Multi-Agent Systems // Conference paper of the IFIP International Federation for Information Processing book series (IFIPAICT, volume 157), Boston, 2004. – pp. 244-246.
4. Jorge J. Gómez-Sanz, Juan Pavón. Methodologies for Developing Multi-Agent Systems // Journal of Universal Computer Science, Graz University of Technology, Austria, 2004. – pp. 370-371.
5. N. V. Malkovskiy, O. N. Granichin, K. S. Amelin. Resource allocation in context of multiagent systems // XII All-Russian conference on management problems, Moscow, 2014. – pp. 9004-9010.
6. M. V. Andreev, A. V. Ivaschenko, P. O. Skobelev. Multiagent system for production resources allocation in heavy engineering // Journal «Software and systems», 2010. – pp. 101-102.

7. E. A. Zinec. Method and means of creating a multiagent system for managing and controlling labor resource allocation // Journal « Scientific works of Vinnitsa National Technical University », 2009. – pp. 1-6.
8. A. S. Nair T. H. Mitch C. D. Flora S. N. Goveas N. Kaabouch P. Ranganathan Multi-Agent Systems for Resource Allocation and Scheduling in a Smart Grid // Journal «Technology and Economics of Smart Grids and Sustainable Energy», Singapore, 2018. – pp. 2-4.
9. D. V. Vins. Multiagent model of the system for operational management of resource allocation on public computing centers // Scientific papers of the conf. of young scientists. Novosibirsk, 2013. – pp. 41-44.
10. Genesis of Knowledge Goals and tasks [Electronic resource]. Available at: http://www.kg.ru/company/about/ (date accessed: 5.05.2022)
11. Carev A. V. Development of MAS Smart Supply Networks for accounting peculiarities of production and transport logistics and possibility of managing large networks of supply with big data processing // Scientific papers of XVIII International conference «Problems of management and modelling in complex systems», Samara, 2016. – pp. 223-236.
12. Yandex.Routing: service of automatic route planning [Electronic resource]. Available at: https://yandex.ru/routing/vrp (date accessed: 5.05.2022)
13. Get started with Yandex.Routing [Electronic resource]. Available at: https://yandex.ru/routing/doc/vrp/ (date accessed: 5.05.2022)

Мягков А. А., Раскатова М. В. Исследование эффективности методов классификации для определения мошеннических банковских транзакций // Международный журнал информационных технологий и энергоэффективности. – 2022. – Т. 7 № 2(24) с. 15–21

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.896

# ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ МЕТОДОВ КЛАССИФИКАЦИИ ДЛЯ ОПРЕДЕЛЕНИЯ МОШЕННИЧЕСКИХ БАНКОВСКИХ ТРАНЗАКЦИЙ

**Мягков А.А., Раскатова М.В.**

*Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ», Россия (111250, г.Москва, ул. Красноказарменная, д.14); e-mail: myagkov.andrey.ru@mail.ru*

**В статье рассматривается исследование работы методов классификации в определении мошеннических банковских транзакций и сравнение полученных результатов. Обработаны исходные данные. Проведен ряд экспериментов по обучению моделей на различных обучающих выборках, полученных в результате балансирования данных, с использованием языка программирования Python. Результаты предсказаний моделей оценены по ряду выбранных метрик.**

Ключевые слова: классификация, транзакция, метрика, машинное обучение, информационная система.

# STUDY OF THE EFFICIENCY OF CLASSIFICATION METHODS FOR IDENTIFYING FRAUDIOUS BANKING TRANSACTIONS

**Myagkov A.A., Raskatova M.V.**

*National Research University “Moscow Power Engineering Institute”, Russia (111250, Moscow, Krasnokazarmennaya street, 14); e-mail: myagkov.andrey.ru@mail.ru*

**The article discusses the study of the work of classification methods in the identification of fraudulent banking transactions and a comparison of the results obtained. Processed initial data. Several experiments were carried out to train models on various training samples obtained because of data balancing using the Python programming language. The results of model predictions are evaluated by several selected metrics.**

Keywords: classification, transaction, metrics, machine learning, information system.

В современном мире большинство операций с деньгами совершаются путём банковских транзакций. Благодаря развитию инфраструктуры приема и обслуживания банковских карт, все больше людей переходит к такому удобному способу оплаты как безналичная оплата. Однако такие транзакции могут быть уязвимы к несанкционированному воздействию третьих лиц, с целью получения доступа к счету и последующему хищению денежных средств. Согласно данным ЦБ, только за 2021 год мошенники похитили 13.5 млрд рублей, что больше, чем в 2020 году (9.7 млрд рублей), а смогли вернуть только 6.8% или 920.5 млн рублей, что меньше, чем в 2020 году (1.1 млрд рублей) [1]. Поэтому важно, чтобы все банковские транзакции подвергались тщательной проверке на предмет мошеннических действий.

Проверка проходящей транзакции происходит в антифрод-системе в процессинговом и авторизационном центрах банка. Там транзакция проверяется на наличие в стоп-листах, проверяется корректность реквизитов и IP-адрес оплаты, чтобы адрес не сильно отличался от обычного и не происходил из стран, где высокий уровень мошеннических действий. Помимо стандартных проверок, в антифрод-системе также используются проверки с помощью машинного обучения, в частности определение мошеннических транзакций с помощью кластеризации и классификации. В данной статье будет рассмотрено определение мошеннических транзакций с помощью бинарной классификации, а именно разделение транзакции на два класса – подлинные и мошеннические, с помощью языка **Python**. В качестве методов классификации были выбраны *дерево решений* (decision tree), *алгоритм k-ближайших соседей* (k-nearest neighbors) и *логистическая регрессия* (logistic regression).

В качестве данных для анализа был выбран датасет «Определение мошеннических транзакций», состоящий из 284807 записей транзакций, сделанных с европейских банковских карт за период сентября 2013 года. У каждой записи 31 поле среди которых:

- Time – количество секунд, прошедших между данной транзакцией и первой транзакцией в наборе данных;
- V1-V28 – параметры без названия, в связи с конфиденциальностью;
- Amount – сумма транзакции;
- Class – переменная, определяющая является ли транзакция мошеннической.

На рисунке 1 показаны первые пять записей из датасета.

| | Time | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | ... | V21 | V22 | V23 | V24 | V2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 | 0.239599 | 0.098698 | 0.363787 | ... | -0.018307 | 0.277838 | -0.110474 | 0.066928 | 0.12853 |
| 1 | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 | -0.078803 | 0.085102 | -0.255425 | ... | -0.225775 | -0.638672 | 0.101288 | -0.339846 | 0.16717 |
| 2 | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 | 0.791461 | 0.247676 | -1.514654 | ... | 0.247998 | 0.771679 | 0.909412 | -0.689281 | -0.32764 |
| 3 | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 | 0.237609 | 0.377436 | -1.387024 | ... | -0.108300 | 0.005274 | -0.190321 | -1.175575 | 0.64737 |
| 4 | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 | 0.592941 | -0.270533 | 0.817739 | ... | -0.009431 | 0.798278 | -0.137458 | 0.141267 | -0.20601 |

Рисунок 1 – Первые пять записей датасета

Из всех записей всего 492 транзакции помечены как мошеннические, что делает этот датасет сильно несбалансированным. Также известно, что столбцы V1-V28 были получены в результате PCA (метод главных компонент), что уже подразумевает в себе использование масштабирования, поэтому для корректного анализа данных были масштабированы оставшиеся столбцы Time и Amount.

Полученные данные были разбиты стратифицированной выборкой, то есть с сохранением распределения классов. Размер тестовой выборки составил 20% от общего числа записей. Для построения и обучения моделей с выбранными методами классификации была использована библиотека **Scikit-learn**.

Для оценки эффективности работы методов классификации используются метрики – специальные показатели, отображающие работу модели. Метрика выбирается относительно поставленной задачи, а неправильно выбранная метрика может привести к заблуждению и неоптимальному решению. Так как в данной задаче бинарная классификация, возможны 4 исхода предсказания модели:

- истинно положительный (TP) – модель предсказала 1 и истинный результат 1;
- ложно положительный (FP) – модель предсказала 1 и истинный результат 0;

- ложно отрицательный (FN) – модель предсказала 0 и истинный результат 0;
- истинно отрицательный (TN) – модель предсказала 0 и истинный результат 1.

В данной задаче были использованы метрики *precision*, *recall* и *F-мера*. Наиболее простая и поэтому распространенная метрика Accuracy, показывающая процент правильно угаданных классов, не использовалась в данной задаче, так как в случае сильно несбалансированных данных она может показывать сильно завышенные результаты, не отображающие способность модели предсказывания миноритарного класса [2].

**Precision** – метрика, показывающая отношение количества истинно положительных результатов (TP) к количеству всех результатов отмеченных положительными моделью (TP + FP), как показано на формуле (1). Данная метрика наиболее ценна в задачах, где ложно положительный результат необходимо минимизировать (положительный результат означает дорогостоящую или долгую проверку).

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

**Recall** – метрика, отображающая отношение количества данных истинно положительных результатов (TP) ко всем результатам, помеченных как положительные (TP + FN), как показано в формуле (2). Данная метрика важна в случаях, когда ценно распознать наибольшее количество всех данных положительного класса (например, определение болезни).

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

Однако зачастую важны как Precision, так и Recall и необходимо найти оптимальный баланс между ними, для этого есть метрика **F-мера** и определяется она по формуле (3).

$$F = (\beta^2 + 1)\frac{Precision * Recall}{\beta^2 * Precision + Recall} \quad (3),$$

где β – переменная, позволяющая отдать большее предпочтение Precision или Recall. При $\beta>1$ приоритет отдается Recall, при $0<\beta<1$ приоритет точности отдается Precision, а при $\beta = 1$ получается частный случай F-меры – мера $F_1$, где равный баланс между Precision и Recall.

Приоритет должен определяться исходя из данных банка, таких как количество сообщений о мошенничестве, количество жалоб о блокировке подлинных транзакций и т. д. Так как в данной задаче такой информации нет, будем считать, что метрики одинаково важны, а основными метриками эффективности модели будут мера F1 и время выполнения.

Работа методов классификации напрямую зависит от их гиперпараметров, поэтому их подбор может улучшить эффективность модели. Так как перебор этих параметров вручную может занять большое количество времени и не обязательно приведет к наилучшему результату, параметры были подобраны с помощью инструмента **GridSearchCV**, который запускает поиск по сетке параметров, т. е. перебирает все возможные значения параметров в выбранном диапазоне и показывает набор параметров, показавший лучшие результаты в выбранной метрике. Оптимизируемыми параметрами для дерева решений выступили -

критерий разделения, максимальная глубина дерева и минимальное количество выборок, для k-ближайших соседей – количество соседей и алгоритм определения ближайших, а для логистической регрессии – метод регуляризации и обратная величина регуляризации.

В таблице 1 показаны результаты работы моделей классификации по выбранным метрикам после подбора гиперпараметров.

Таблица 1 – Таблица значений моделей классификации по выбранным метрикам

| | **Значение Precision среди мошеннических** | **Значение Recall среди мошеннических** | **Значение $F_1$ среди мошеннических** | **Время выполнения** |
|---|---|---|---|---|
| Дерево решений | 0.8652 | 0.7857 | 0.8235 | 4.30 |
| Алгоритм k-ближайших соседей | 0.9494 | 0.7653 | 0.8475 | 4:29.45 |
| Логистическая регрессия | 0.8636 | 0.5816 | 0.6951 | 2.64 |

Так как несбалансированные данные могут влечь за собой ошибки, связанные с тем, что обученные модели будут предполагать, что в основном транзакции являются не мошенническими, были рассмотрены методы для корректировки распределения классов, такие как Недостаточная выборка (Undersampling) и Передискретизация (Oversampling). Стоит понимать, что корректировка распределения происходит только для обучающей выборки, так как проверять работу модели мы хотим на изначальных данных.

В случае недостаточной выборки данные мажоритарного класса удаляются до тех пор, пока не будет достигнут нужный баланс. Одним из способов недостаточной выборки является алгоритм **NearMiss**, в котором удаляются те данные мажоритарного класса, которые наиболее близко находятся к данным миноритарного класса, тем самым делая большую разницу между классами для последующей классификации. Применения алгоритма NearMiss осуществляется с помощью одноименной функции входящей в состав пакета функций under_sampling библиотеки imblearn.

При передискретизации же балансировка происходит путем добавления данных миноритарного класса. Наиболее простой способ – дублирование существующих данных, что хоть и сбалансирует данные, но не даст никаких новых данных для обучения модели. Улучшением данного способа является технология **SMOTE** (Synthetic Minority Oversampling Technique), которая генерирует новые данные похожие на те, что уже есть в датасете. Применение алгоритма SMOTE осуществляется с помощью одноименной функции из пакета функций over_sampling.

Для моделей, обученных на сбалансированных данных, также были определены и применены оптимальные гиперпараметры с помощью GridSearchCV. Результаты работы моделей классификации, обученных на сбалансированных данных, полученных в результате работы методов NearMiss и SMOTE показаны в таблицах 2 и 3 соответственно.

Таблица 2 - Таблица значений моделей классификации по выбранным метрикам при NearMiss

| | **Значение Precision среди мошеннических** | **Значение Recall среди мошеннических** | **Значение $F_1$ среди мошеннических** | **Время выполнения** |
|---|---|---|---|---|
| Дерево решений | 0.0053 | 0.9286 | 0.0105 | 0.02 |
| Алгоритм k-ближайших соседей | 0.0118 | 0.9082 | 0.0233 | 2.15 |
| Логистическая регрессия | 0.0094 | 0.9184 | 0.0186 | 0.04 |

Таблица 3 - Таблица значений моделей классификации по выбранным метрикам при SMOTE

| | **Значение Precision среди мошеннических** | **Значение Recall среди мошеннических** | **Значение $F_1$ среди мошеннических** | **Время выполнения** |
|---|---|---|---|---|
| Дерево решений | 0.0240 | 0.9082 | 0.0467 | 5.50 |
| Алгоритм k-ближайших соседей | 0.5743 | 0.8673 | 0.6911 | 8:38.74 |
| Логистическая регрессия | 0.0608 | 0.9286 | 0.1142 | 5.77 |

Из полученных результатов по метрике $F_1$ видно, что модель, построенная при использовании технологии SMOTE, работает лучше, чем при NearMiss, но намного хуже, чем при изначальных несбалансированных данных. В данном случае балансировка данных не улучшила результат и использовать ее не стоит.

Следующим шагом была попытка улучшить результаты по методу логистической регрессии путем его модификации. Логистическая регрессия – разновидность множественной регрессии, использующая логистическую функцию для моделирования зависимости бинарной выходной переменной от набора входных данных. Не стоит путать с регрессионным алгоритмом, так как логистическая регрессия — это алгоритм классификации [3]. Классификация логистической регрессии работает путем подсчета вероятности принадлежности одному из классов. В случае бинарной классификации вероятность того, что значение принадлежит к одному классу (например, что транзакция мошенническая) обозначим P+, а вероятность того, что значение принадлежит ко второму классу (например, транзакция подлинная) P-. Тогда P+ = 1 - P-, а сами вероятности лежат в диапазоне [0, 1]. Определение класса происходит путем выбора наибольшей вероятности, так в случае бинарной классификации пороговым значением является 0.5.

Однако такое пороговое значение, может быть, не всегда целесообразно и в определенных случаях результаты по метрике $F_1$ бы улучшились изменой порога. Например, в случае результатов начальных несбалансированных данных видно, что у модели показатели метрики Precision выше, чем у метрики Recall и более высоких результатов метрики $F_1$ можно добиться, снизив порог.

Так как у базового класса логистической регрессии нет возможности установки своего порогового значения, бы написан класс, расширяющий возможности базового. В классе переопределен метод предсказания так, чтобы можно было устанавливать новое пороговое значение и написан метод помогающий определить оптимальное пороговое значение из

тренировочных данных, дающее максимальное значение по метрике f1. Результат работы улучшенной логистической регрессии по метрике $F_1$ составил 0.7568, что лучше результатов обычной логистической регрессии на 6%, получилось это путем понижения Precision и повышения Recall.

В случаях, когда модель обучается на тренировочных данных и оптимизирует под них параметры модели, есть вероятность, что модель слишком хорошо научится определять тренировочные данные, а тестовые данные будет определять намного хуже [4]. Для проверки корректности работы построенных моделей на предмет переобучения была произведена кросс валидация, суть ее заключается в следующем:

- Разделить тренировочные данные на n непересекающихся одинаковых по объему частей;
- Обучить модель на k-1 частей;
- Протестировать на оставшейся части;
- Повторить k раз, каждый раз выбирая новую часть для проверки.

Разбиение тренировочных данных происходило с использованием метода **StratifiedKFold**, который разбивает данные на k равных частей с одинаковым распределением классов в них. Результаты кросс валидации основных моделей показали похожие значения на полученные у моделей ранее, а разброс значений везде не превышал 2%, что хороший показатель и означает, что переобучение не происходит.

Из проделанного исследования можно сделать вывод, что наилучшей моделью для определения мошеннических транзакций является модель, построенная с использованием дерева решений при несбалансированных данных с использованием оптимальных гиперпараметров. Хотя метод k-ближайших на тех же данных по метрике $F_1$ и показал на 2% больший результат, время, затраченное на обучение и предсказание, занимает 4.5 минуты, что несоизмеримо больше, чем у дерева решений.

**Список литературы**

1. Годовой отчет Банка России за 2021 год – URL: https://www.cbr.ru/Collection/Collection/File/40915/ar_2021.pdf (дата обращения: 27.11.2021). – Режим доступа: открытая информация. – Текст: электронный
2. Бенджио, И. Глубокое обучение / И. Бенджио, Я. Гудфеллоу, А. Курвилль – ДМК Пресс, 2017. – 652с.
3. Шолле, Ф. Глубокое обучение на Python / Ф. Шолле – Питер, 2018. – 400с.
4. Орельен, Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем / Ж. Орельен, 2018. – 688 с.

**References**

1. Bank of Russia Annual Report for 2021 – URL: https://www.cbr.ru/Collection/Collection/File/40915/ar_2021.pdf (date of access: 11/27/2021). – Access mode: open information. – Text: electronic
2. Bengio, I. Deep Learning / I. Bengio, Y. Goodfellow, A. Courville - DMK Press, 2017. - 652p.
3. Chollet, F. Deep learning in Python / F. Chollet - Peter, 2018. - 400p.

4. Aurelien, J. Applied Machine Learning with Scikit-Learn and TensorFlow. Concepts, tools and techniques for creating intelligent systems / J. Aurelien, 2018. - 688 p.

Лашков А.А., Зернов М.М. Способ и алгоритмы поиска маршрутов протяженных объектов на цифровой карте местности// Международный журнал информационных технологий и энергоэффективности. – 2022
Т. 7 № 2(24) с. 22–31

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 681.3.019

# СПОСОБ И АЛГОРИТМЫ ПОИСКА МАРШРУТОВ ПРОТЯЖЕННЫХ ОБЪЕКТОВ НА ЦИФРОВОЙ КАРТЕ МЕСТНОСТИ

**[1]Лашков А.А., [2]ЗерновМ.М.**
*Филиал ФГБОУ ВО «Национальный исследовательский университет «МЭИ» в г. Смоленске, Россия, (214013, г. Смоленск, Энергетический проезд, 1), e-mail: [1]lahkandr010@mail.ru, [2]zmmioml@yandex.ru*

**Данная статья посвящена рассмотрению возможных путей решения задачи поиска маршрута протяженного объекта с использованием навигационного графа. Предложены этапы и особенности реализации способа поиска маршрутов протяженных объектов на цифровой карте местности. Выделены основные этапы поиска маршрутов на цифровой карте местности. Дано описание формирования представления цифровой карты местности с учетом коэффициента проходимости объектов этой карты. Дано описание и формирование навигационного графа по скелету свободной области цифровой карты местности. Описано получение скелета свободной области из диаграммы Вороного, построенного на основе сегментного представления препятствия. В роли препятствий выступают объекты цифровой карты местности, которые соответствуют выбранному коэффициенту проходимости. Сформулирован модифицированный способ поиска кратчайшего пути на графах A-star.**

Ключевые слова: навигационный граф, КД-дерево, скелет, диаграмма Вороного.

# METHOD AND ALGORITHMS FOR FINDING ROUTES OF EXTENDED OBJECTS ON A DIGITAL TERRAIN MAP

**[1]Lashkov A.A., [2]Zernov M.M.**
*[1,2]Smolensk Branch of the National Research University "Moscow Power Engineering Institute", Smolensk, Russia (214013, Smolensk, Energetichesky proezd, 1), e-mail: [1]lahkandr010@mail.ru, [2]zmmioml@yandex.ru*

**This article is devoted to the consideration of possible ways to solve the problem of finding the route of an extended object using a navigation graph. The stages and features of the implementation of the method of searching for routes of extended objects on a digital terrain map are proposed. The main stages of route search on the digital map of the area are highlighted. The description of the formation of the representation of a digital terrain map is given, taking into account the coefficient of patency of the objects of this map. The description and formation of the navigation graph on the skeleton of the free area of the digital terrain map is given. It is described how to obtain a skeleton of a free area from a Voronoi diagram constructed on the basis of a segmental representation of an obstacle. The objects of the digital terrain map that correspond to the selected cross-country coefficient act as obstacles. A modified method for finding the shortest path on A-star graphs is formulated.**

Keywords: navigation graph, KD-tree, skeleton, Voronoi diagram.

## Введение

Данная работа является продолжением статьи «Применение навигационного графа для решения задачи поиска маршрута габаритного объекта по цифровой карте местности» [1]. По результатам анализа способов и алгоритмов построения маршрута на цифровой карте

местности(ЦКМ), рассмотренных в статьях [3-6], можно выделить основные этапы поиска маршрутов на ЦКМ, описанные ниже.

1. Формирование представления карты.

Включает в себя этапы обработки карты такие как:

- выбор объектов на основе характеристик (площади, проходимости, типа и т.д.);
- корректировка объектов (например, обработка самопересечение площадных объектов);
- упрощение карты (аппроксимация границ объектов, объединение объектов и т.д.).

2. Формирование графа для поиска пути.

Чаще всего для поиска пути используется граф, например, граф контрольных точек(waypoints)[7], граф видимости[3] и т.д.

3. Нахождение пути на графе.

Существует множество алгоритмов писка пути на графе, наиболее известны алгоритмы Дейкстры и A-star [8].

4. Оптимизация маршрута.

Данный этап заключается в уточнение маршрута, например в уменьшении протяженности, уменьшении или увеличении количества узлов в зависимости от точности и т.д.

**Способ и алгоритмы поиска маршрутов протяженных объектов на цифровой карте местности**

На основе вышеперечисленных пунктов предлагается способ поиска маршрутов протяженных объектов на цифровой карте местности, состоящий из следующих этапов.

1. Формирование карты проходимости на основе ЦКМ, выделение свободных областей и препятствий. Формирование карты препятствий.
2. Формирование скелета свободной области на основе диаграммы Вороного и выборка вершин данной диаграммы, не лежащих на препятствиях.
3. Формирование навигационного графа, позволяющей охарактеризовать свободную область. Выбор вершин для графа из диаграммы Вороного, на основе расстояния до ближайшего препятствия, и их соединение.
4. Поиск кратчайшего пути с учетом ширины объекта на основе алгоритма A-star во взвешенном графе, адаптированного к использованию оценки расстояния до препятствия от вершины навигационного графа.
5. Улучшение построенного пути за счет удаления лишних вершин и итерационного уточнения пути.

**Формирование карты проходимости на основе ЦКМ, выделение свободных областей и препятствий. Формирование карты препятствий**

Начальный этап заключается в формировании карты проходимости на основе ЦКМ, выделении свободных областей и препятствий. Необходимо преобразовать ЦКМ в форму удобную для построения навигационного графа, а для этого нужно выделить область, занимаемую препятствиями (бездорожьем) и свободную область. В результате решения,

данной задачи формируется список объектов карты, представляющие собой многоугольники со своими коэффициентами проходимости, изменяемые от 0(свободен) до 1(непроходим).

Для этого различным типам объектов ЦКМ (слоям) должны быть сопоставлены значения коэффициентов проходимости, т.е. заранее составлены справочники с коэффициентами проходимости для каждого объекта. Опираясь на данные справочника, составляются списки объектов-препятствий (например, выбрать все объекты как препятствия, значения коэффициента проходимости которых больше 0.5).

После того как выбраны объекты-препятствия необходимо создать карту препятствий (набор непересекающихся многоугольников). Так как цифровая карта местности может содержать самопересечение объектов и пересечение объектов между собой, то необходимо заменять каждый многоугольник с самопересечением на внешний контур или разделять на несколько многоугольников, и объединять пересекающиеся объекты. Данные действия способствуют корректному построению скелета свободной области.

Таким образом, можно предложить следующий алгоритм формирования карты препятствий на основе ЦКМ.

Итак, есть типы перемещаемых объектов (человек, колёсный транспорт, гусеничный транспорт, и т.д.). Каждому типу перемещаемого объекта для каждого типа объектов карты сопоставлена своя проходимость. У каждого типа перемещаемого объекта свой порог проходимости.

1. Установка верхней границы коэффициента проходимости, в зависимости от типа перемещаемого объекта.
2. Т.к. ЦКМ имеет разные типы объектов(леса, водоемы, кварталы и т. д.) сгруппированные по слоям, то формируются наборы объектов для каждого слоя.
3. В каждой группе удаляются объекты, коэффициент которых больше установленного порога для указанного типа объекта, т.е. «проходимые» объекты.
4. Проход по каждой группе и удаление самопересечения(рисунок 1.а и 1.б)
5. Объединение в каждой группе объектов, которые пересекаются (рисунок 1.в).
6. Объединение объектов между группами, также сопровождается устранением пересечений.

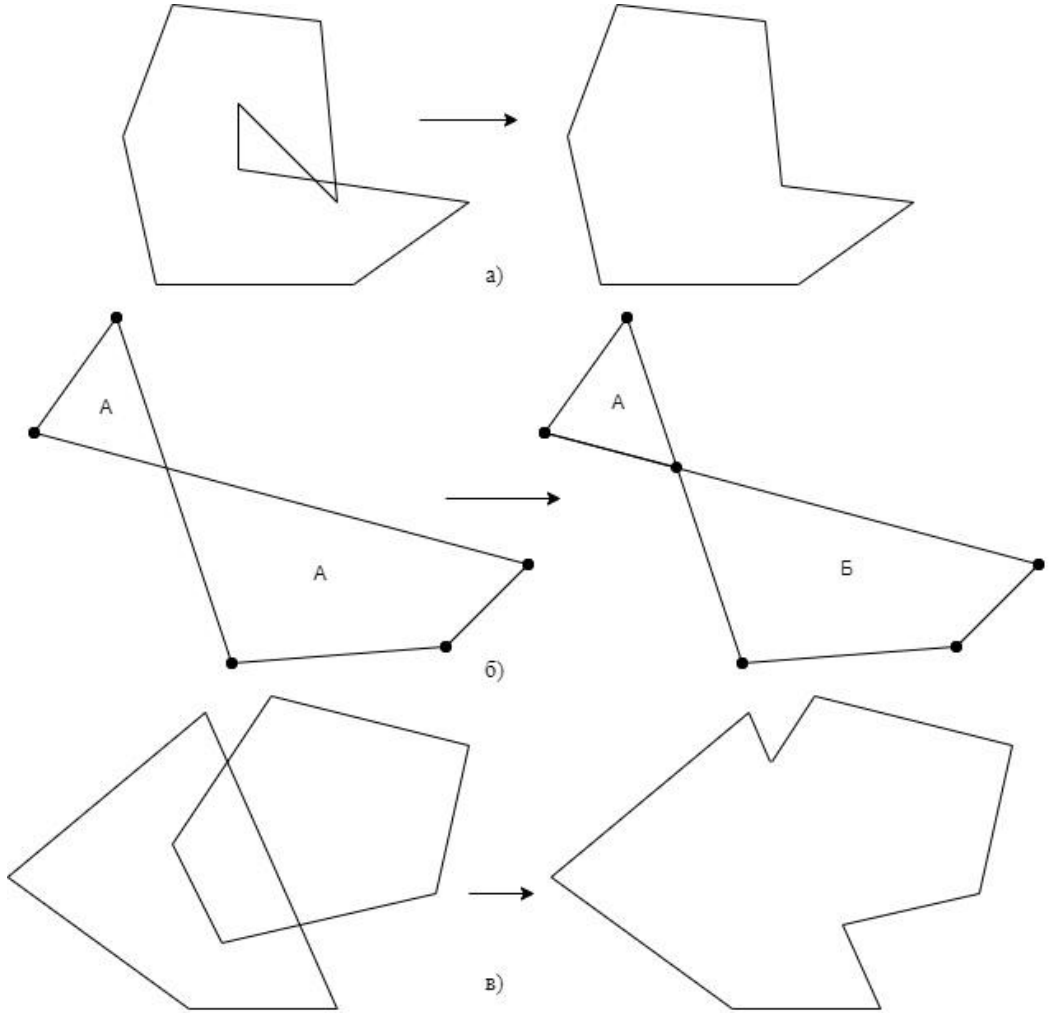


Рисунок 1 – Операции устранения самопересечения и операция объединения пресекающихся объектов(***а*** – замена многоугольника с самопересечением на его контур, ***б*** – деление многоугольника с самопересечением на несколько многоугольников, ***в*** – объединение пресекающихся объектов).

Операции устранения самопересечения и операция объединения пресекающихся объектов выполняются в соответствии с алгоритмом, рассмотренным в [2]. Таким образом получаем набор непроходимых непересекающихся многоугольников, который представляет карту препятствий.

**Формирование скелета свободной области**

Строим скелет карты препятствий на основе Диаграммы Вороного. Используем алгоритм построения для сегментов[3], т.е. необходимо получить из набора препятствий(карты препятствий) набор сегментов, для чего достаточно обойти каждый многоугольник объекта-препятствия и сформировать сегменты. Ребра диаграммы Вороного используются в качестве элементов скелета (рисунок 2).

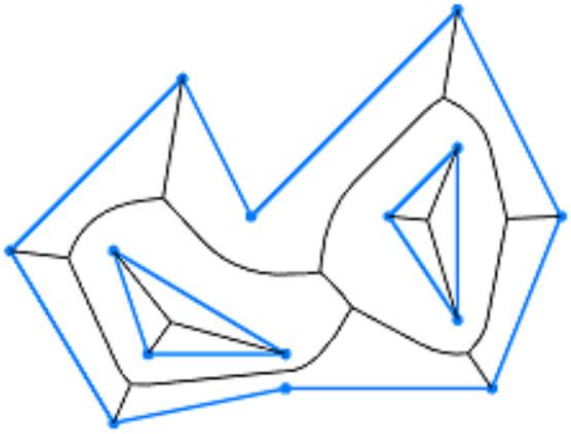

Рисунок 2 – Пример скелета (черные линии)

В результате построения ребра диаграммы Вороного будут лежать, как внутри препятствий, так и на внешней совокупности свободных областей, значит следующим шагом будет определение ребер диаграмм Вороного, которые лежат вне препятствий, т.е. на свободной области, такие ребра и будут являться скелетом карты препятствий. Т.к. все полученные ребра диаграммы Вороного формируют набор компонент связности, то достаточно будет начать проходить все узлы таких графов и при этом определять, лежат ли они на препятствии или нет. Допустим $v$ – какой-то узел, если $v$ лежит на препятствии, то обходим граф и помечаем все узлы данного графа, как узлы, которые лежат на препятствии, тоже самое делаем и если $v$ лежит на свободной области.

Для каждого из оставшихся узлов находим минимальное расстояние до ближайшего препятствия. Пусть - узел $v_i$ с наибольшим минимальным расстоянием $l_m$, убираем узел $v_i$ из рассмотрения и все узлы, которые покрывает окружность с центром в узле $v_i$ и радиусом $l_m$, после чего переходим к следующему нерассмотренному узлу. Повторяем до тех пор, пока узлы не закончатся. Далее формируем новые связи(ребра) между выбранными узлами на основе старых связей, т.е. допустим была последовательность связанных узлов $v_i, v_{i+1}, v_{i+2}, \dots, v_{i+n}$, допустим $v_i$ и $v_{i+n}$ имеют наибольшие минимальные расстояние до ближайшего препятствия и покрывают узлы $v_{i+1}, v_{i+2}, \dots, v_{i+n-1}$, то формируем связь между $v_i$ и $v_{i+n}$.

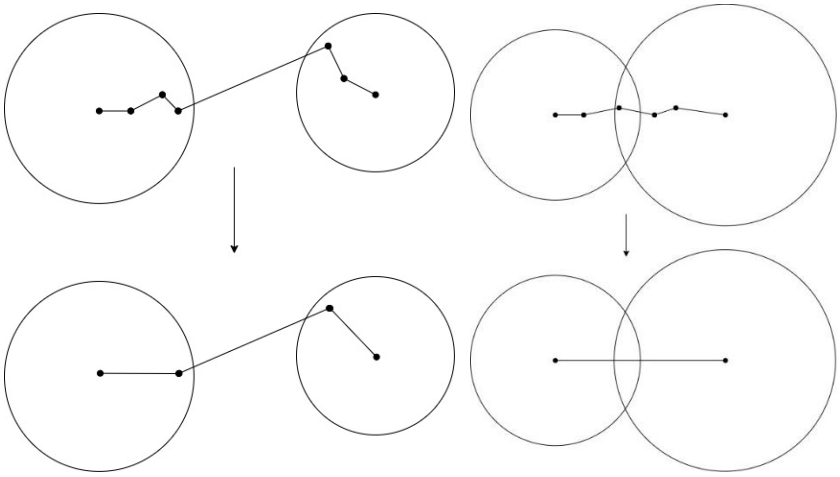

Рисунок 3 – Устранение покрытых вершин окружностями вершин

Как видно из рисунка 3, если окружности пересекаются, то вершины, являющиеся центрами этих окружностей, соединяются одним ребром, иначе маршрутом из 3-х. Таким образом уменьшается количество узлов и образуется навигационный граф, который используется для построения итогового маршрута.

**Поиск кратчайшего пути с учетом ширины объекта на основе алгоритма A-star**

По сформированному навигационному графу ищем путь, используя алгоритм поиска кратчайшего пути на графе A-star с учетом ширины объекта. Используется стандартный алгоритм A-star, но с дополнительной проверкой на то пройдет ли данный объект по ребру или нет. Формируется описывающий прямоугольник вокруг ребра, т.е. ребро как бы «раздувается» (рисунок 4) во все стороны на половину длины ширины объекта, а после проверяется пересекается ли построенный многоугольник с ближайшими препятствиями, если да, то он подходит для дальнейшего рассмотрения в алгоритме A-star, иначе убираем данное ребро из рассмотрения. Для быстрого нахождения ближайшего препятствия можно построить КД-дерево[9], позволяющее ускорить поиск ближайшего препятствия к точке. Таким образом формируется путь для объекта с определённой шириной, по которому он может перемешаться, не задевая препятствий.

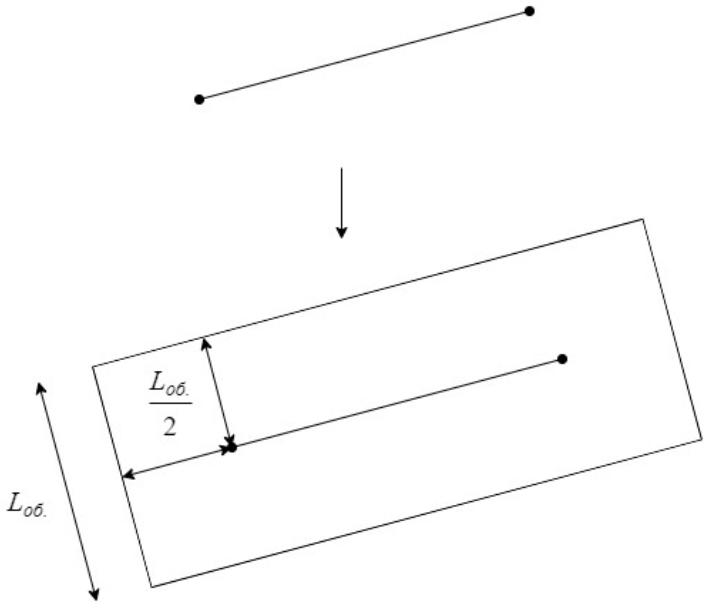


Рисунок 4 – «Раздутие» ребра

Модифицированный алгоритм поиска маршрута имеет шаги, описанные ниже.

Обозначим веса вершин:

- G - стоимость передвижения из начальной вершины к данной вершине, следуя найденному пути к данной вершине;
- F – стоимость равная сумме G и функции h(x) для данной вершины.

Шаги алгоритма A*:

1) Формирование «открытого» списка (в начале работы алгоритма содержит начальную вершину), в котором находятся вершины, которые необходимо просмотреть, и «закрытого» списка, который содержит просмотренные вершины.

2) Выбрать в качестве текущей вершины «C» вершину с наименьшим значением F и поместить в закрытый список.

3) Нахождение всех соседних вершин «N», который не входят в закрытый список, и соответствующих инцидентных ребер «E».

4) Каждое ребро «раздувается» (рисунок 4) и проверяется пресечение с ближайшими препятствиями, если пересечения есть, то убираем из рассмотрения соответствующую соседнюю вершину «N», т.е. помещаем ее в закрытый список.

4) Расчет для каждого оставшегося в рассмотрении соседа «N» временного G_TEMP равное сумме веса G текущей вершины «C» и веса ребра «C»-«N». Если «N» находится в открытом списке и его текущее G больше временного G_TEMP, или «N» не находится в открытом списке, то соответственно обновляем в отрытом списке или добавляем в отрытый список вершину «N», где G = G_TEMP и рассчитываем F, еще указываем для «N» вершину, из которой пришли(«C»).

5) Необходимо повторить описанный выше шаги с шага 2 до тех пор, пока не будет посещен пункт назначения.

После нахождения кратчайшего пути применяем алгоритмы улучшения пути за счет удаления лишних вершин и итерационного уточнения траектории пути, рассмотренные в [4].

Для каждой вершины $v_i$ проверяется возможность создания ребра между вершиной $v_i$ и $v_{i+2}$. Формируем ребро и определяем, лежит ли оно на расстоянии в половину ширины перемещаемого объекта от ближайших препятствий, если нет, то удаляется вершина $v_{i+1}$, соединяются ребром вершины $v_i$ и $v_{i+2}$, и следующей для рассмотрения берется вершина $v_{i+2}$, если да, сохраняется вершина $v_{i+1}$ и рассматривается как следующая вершина для обработки. Данные действия продолжаются пока количество вершин пути уменьшается.

Дальнейшем шагом будет сокращение длины пути путем как можно близкого огибания препятствия («натягивание пути»). Допустим берем вершину $v$ на кратчайшем пути, берем инцидентные для выбранной вершины ребра $e_1$ и $e_2$. Вдоль ребер $e_1$ и $e_2$ от вершины $v$ откладываем заранее найденный шаг h, при этом получаем две вершины $v_{1(1h)}$ и $v_{2(1h)}$ на соответствующих ребрах. Далее проверяется возможность создания ребра между вершинами $v_{1(1h)}$ и $v_{2(1h)}$, как это было описано при удалении избыточных вершин, если вершины можно соединить, то продолжаем откладывать вдоль ребер $e_1$ и $e_2$ вершины $v_{1(2h)}$ и $v_{2(2h)}$ с шагом 2h и также пытаемся их соединить. Процесс продолжается до тех пор, пока мы можем соединять вновь отложенные вершины, или пока не будет достигнута конечная точка одного из ребер. После чего мы заменяем вершину $v$ последними вершинами $v_{1(ih)}$ и $v_{2(ih)}$, которые прошли проверку на соединение между собой, и переходим к следующей вершине, если таких вершин нет, то сохраняем вершину $v$ и переходим к следующей вершине. Данные действия продолжаются до тех пор, пока вершины заменяются, как только замены прекращаются, уменьшается шаг h, после чего продолжается выполнение алгоритма с новым шагом, пока шаг не достигнет определенного допустимого значения (устанавливается пользователем или рассчитывается, изначальный шаг можно взять как среднее арифметическое минимальных расстояний вершин пути до препятствий деленный на коэффициент). После чего повторяем алгоритм удаления избыточных вершин.

В итоге получаем следующий набор шагов, изображенный на рисунке 5.

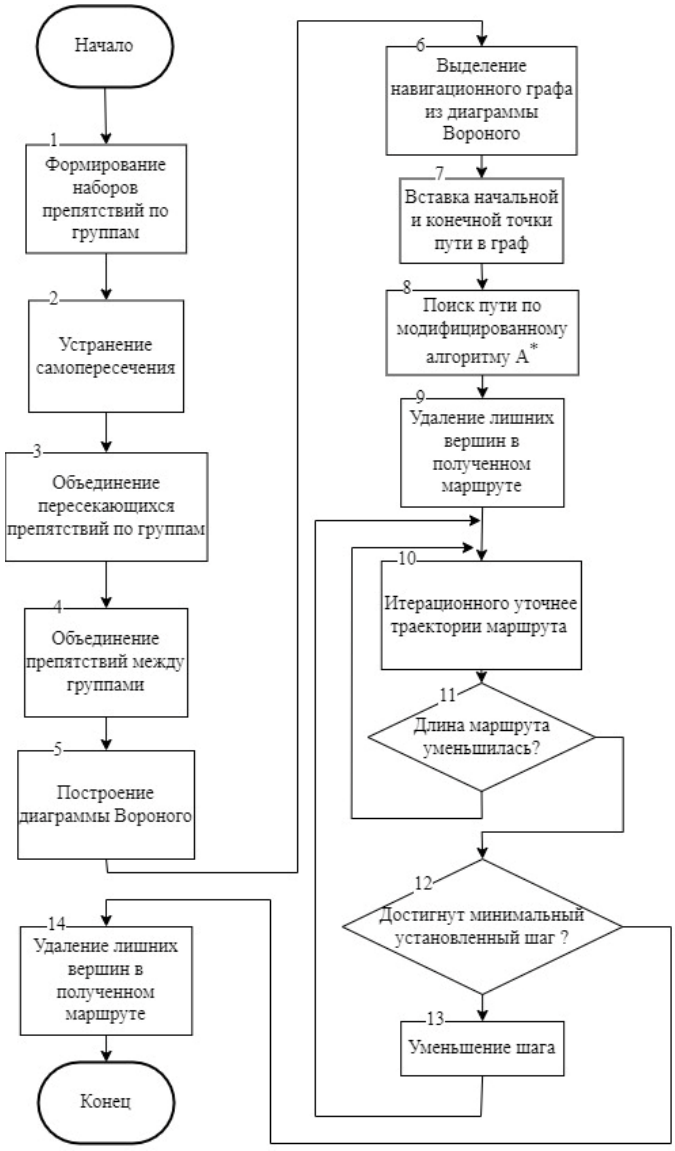


Рисунок 5 – Способ поиска маршрутов протяженных объектов на цифровой карте местности

**Заключение**

Предложен способ поиска маршрутов протяженных объектов на цифровой карте местности, характеризующиеся следующими особенностями:

- поиск маршрута осуществляется на цифровой карте местности с учетом коэффициента проходимости объектов;
- учет ширины объекта при формировании маршрута;
- применением навигационного графа, построенного на основе скелета проходимых участков;
- применением модифицированного алгоритма поиска пути на графах A-star;
- алгоритмами уточнения траектории построенного маршрута.

**Список литературы**


1. Лашков А.А., Зернов М.М. Применение навигационного графа для решения задачи поиска маршрута габаритного объекта по цифровой карте местности // Международный журнал информационных технологий и энергоэффективности, [S.l.], v. 7, n. 1(23), С. 3-16, апр. 2022. ISSN 2500-1752
2. THE BOOST.POLYGON LIBRARY [Электронный ресурс] URL: https://www.boost.org/doc/libs/1_66_0/libs/polygon/doc/index.htm (дата обращения 15.05.22)
3. Mark B. et al. Computational geometry algorithms and applications. – Spinger, 2008.
4. Bhattacharya P., Gavrilova M. L. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path //IEEE Robotics & Automation Magazine. – 2008. – Т. 15. – №. 2. – С. 58-66.
5. Pettré J., Grillon H., Thalmann D. Crowds of moving objects: Navigation planning and simulation //ACM SIGGRAPH 2008 classes. – 2008. – С. 1-7.
6. Pettré J. et al. Real-time navigating crowds: scalable simulation and rendering //Computer Animation and Virtual Worlds. – 2006. – Т. 17. – №. 3-4. – С. 445-455.
7. Toma A. I. et al. Waypoint Planning Networks //arXiv preprint arXiv:2105.00312. – 2021.
8. Rachmawati D., Gustin L. Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem //Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1566. – №. 1. – С. 012061.
9. Анатомия KD-Деревьев [Электронный ресурс] URL: https://habr.com/ru/post/312882/ (дата обращения 5.05.22)


**References**


1. Lashkov A.A., Zernov M.M. Application of the navigation graph to solve the problem of finding the route of a dimensional object on a digital terrain map // International Journal of Information Technology and Energy Efficiency, [S.L.], v. 7, No. 1(23), pp 3-16, Apr. 2022. ISSN 2500-1752
2. THE BOOST.POLYGON LIBRARY [E-resource] URL: https://www.boost.org/doc/libs/1_66_0/libs/polygon/doc/index.htm (Address date 15.05.22)
3. Mark B. et al. Computational geometry algorithms and applications. – Spinger, 2008.
4. Bhattacharya P., Gavrilova M. L. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path //IEEE Robotics & Automation Magazine. – 2008. – Т. 15. – №. 2. – pp. 58-66.
5. Pettré J., Grillon H., Thalmann D. Crowds of moving objects: Navigation planning and simulation //ACM SIGGRAPH 2008 classes. – 2008. – pp 1-7.
6. Pettré J. et al. Real-time navigating crowds: scalable simulation and rendering //Computer Animation and Virtual Worlds. – 2006. – Т. 17. – №. 3-4. – pp. 445-455.
7. Toma A. I. et al. Waypoint Planning Networks //arXiv preprint arXiv:2105.00312. – 2021.
8. Rachmawati D., Gustin L. Analysis of Dijkstra's Algorithm and A* Algorithm in Shortest Path Problem //Journal of Physics: Conference Series. – IOP Publishing, 2020. – Т. 1566. – №. 1. – pp. 012061.
9. Anatomy of KD-Trees [E-resource] URL: https://habr.com/ru/post/312882 / (Address date 5.05.22)

Тимешов А.С., Раскатова М.В. Формирование модели генетического алгоритма для решения задачи составления расписания занятий в школе // Международный журнал информационных технологий и энергоэффективности. – 2022. – Т.7 №2(24) с.32-39

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.896

# ФОРМИРОВАНИЕ МОДЕЛИ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗАДАЧИ СОСТАВЛЕНИЯ РАСПИСАНИЯ ЗАНЯТИЙ В ШКОЛЕ

**Тимешов А.С., Раскатова М.В.**
*Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ», Россия (111250, г.Москва, ул. Красноказарменная, д.14); e-mail: chel.ed@yandex.ru*

**В статье рассматривается проектирование модели генетического алгоритма, решающего задачу составления расписания занятий, и ее программная реализация. Программная реализация написана с использованием языка программирования Python и библиотеки Deap.**

Ключевые слова: генетические алгоритмы, модель алгоритма, расписание занятий, школа, библиотека Deap

# A MODEL OF A GENETIC ALGORITHM FORMATION FOR SOLVING THE PROBLEM OF SCHOOL SCHEDULES

**Timeshov A.S., Raskatova M.V.**
*National Research University “Moscow Power Engineering Institute”, Russia (111250, Moscow, Krasnokazarmennaya street, 14); e-mail: chel.ed@yandex.ru*

**The article discusses the design of a genetic algorithm model that solves the problem of scheduling classes, and its software implementation. The software implementation is written using the Python programming language and the Deap library.**

Keywords: genetic algorithms, algorithm model, decision evaluation strategy, school schedule, school, Deap library.

Человечество на протяжении всей своей истории сталкивалось с необходимостью составления расписаний. Под расписанием обычно подразумевается набор некоторых действий, выполнение которых нужно расставить в определенный промежуток времени, учитывая разнообразные ограничения, накладываемые на порядок выполнения этих действий [1].

Каждый человек, зная, к какому сроку требуется выполнить определенное действие и, предполагая, как много времени и ресурсов потребуется на его выполнение, занимается планированием своего личного расписания. Чаще всего составление такого расписания не вызывает трудностей. Они возникают лишь в том случае, когда задач становится много, когда появляется необходимость принять во внимание множество дополнительных факторов и условий, когда появляется необходимость составления расписания не для одного человека, а

для целого коллектива[2]. Таким коллективом могут быть участники учебного процесса в учебном заведении - преподаватели и учащиеся.

Составление расписания занятий в учебном заведении является очень трудоёмкой задачей, потому как интересы участников учебного процесса многообразны, а все факторы, которые могут повлиять на расписание, практически невозможно учесть. Поэтому правильно подобранный алгоритм составления расписания позволит наиболее рационально и эффективно распределить рабочее время и ресурсы учебного процесса. Некоторыми из возможных вариантов решения подобной задачи являются применения таких алгоритмов как: алгоритм метода раскраски графа, алгоритм градиентного спуска, жадные алгоритмы и генетические алгоритмы.

Генетический алгоритм – это последовательность управляющих действий и операций, имитирующая эволюционные процессы в природе такие как: скрещивание, естественный отбор и мутация, для решения задач поиска и оптимизации.

В генетическом алгоритме потенциальное решение задачи, в данном контексте, это расписание занятий, представляется хромосомой, т.е. в закодированном виде. И каждая такая хромосома несет в себе набор генов, др. словами элементы решения или его свойства.

Генетический алгоритм постепенно развивает популяцию, то есть набор потенциальных решений задачи. Эти решения называются индивидуумами или особями, они итеративно оцениваются и используются для создания нового поколения индивидуумов [3].

Особи, которые получили наивысшую оценку, с большей вероятностью могут пройти отбор и, скрещиваясь с другими особями, передать свои свойства будущему поколению. Таким образом, потенциальные решения задачи постепенно эволюционируют и в конечном итоге, находится лучшее решение [4]. Периодически случайным образом популяция обновляется, меняются сочетания генов в хромосомах, то есть происходит мутация. Целью мутации является стимулирование алгоритма на поиск неисследованных областей потенциальных решений.

Общую схему работы генетического алгоритма можно представить в виде схемы, приведенной на рисунке 1.


Рисунок 2 - Общая схема работы генетического алгоритма

В каждом учебном заведении можно выделить следующие множества объектов:

- Учебные классы (множество C = $\{c_0, c_1, \ldots, c_{k-1}\}$, где $c_i$ – учебный класс, $0 \leq i \leq k-1$ , k - количество учебных классов в школе).
- Учителя (множество T = $\{t_0, t_1, \ldots, t_{l-1}\}$, где $t_i$ - учитель, $0 \leq i \leq l-1$ , $l$ - количество учителей в школе).
- Дисциплины (множество D = $\{d_0, d_1, \ldots, d_{m-1}\}$, где $d_i$ - дисциплина, $0 \leq i \leq m-1$ , m – количество дисциплин, преподаваемых в школе).
- Аудитории (множество A = $\{a_0, a_1, \ldots, a_{n-1}\}$, где $d_i$ - дисциплина, $0 \leq i \leq n-1$ , n – количество аудиторий в школе).

- Временные интервалы проведения учебных занятий (уроков) (множество L = $\{l_0, l_1, \ldots, l_{p-1}\}$, где $l_i$ – временной интервал урока, $0 \leq i \leq p-1$, p – количество уроков, доступных для проведения занятий за одну неделю для всей школы).

Чтобы решить задачу, важно определить ключевые множества, для которых будет строиться решение, и которые обязательно нужно включить в структуру гена.

В случае если в учебном заведении каждому учителю присвоена отдельная аудитория, а наименование дисциплин не имеет значения для составления расписания, важно только сопоставить учителей и учебные классы в определенный момент времени, то в качестве обязательных множеств, необходимых для построения гена, можно выбрать учителей, учебные классы и уроки.

Каждому классу, учителю и уроку может быть присвоен свой номер. Таким образом, формируя каждый ген в виде последовательности {T, C, L}, где T - номер учителя, C - номер класса, L - номер урока, получается следующая модель хромосомы, представленная на рисунке 2.

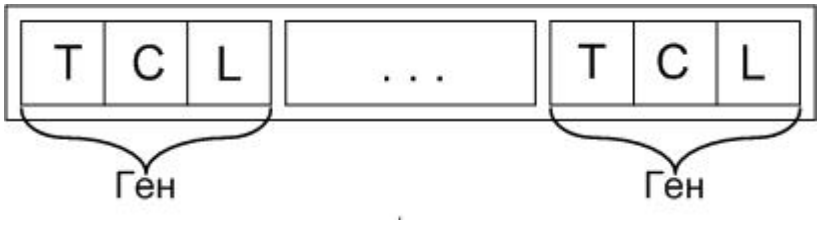


Рисунок 2 - Модель хромосомы

В данной хромосоме каждый ген несет в себе всю необходимую информацию для его корректного декодирования, а его расположение в хромосоме не имеет абсолютно никакого значения, что позволяет применить для данной модели большее число возможных моделей скрещивания.

Начальная популяция решений является отправной точкой, откуда начинается поиск решения. От того, насколько удачно она составлена, может зависеть время, которое будет затрачено на поиск, и качество конечного результата. Поскольку связь между учителем и классом является статической и не может каким-либо образом разрываться или меняться в процессе всей работы алгоритма, то единственной составляющей гена, которая может хоть как-то изменяться, является номер урока.

В общем случае количество допустимых уроков в неделю (которое можно выделить на проведение учебных занятий) для каждого класса в общеобразовательном учреждении равно 40 (8 уроков на каждый будний день), то для начальной популяции достаточно будет создать расписания, в каждом из которых будет только один из сорока возможных номеров урока для всех генов. Очевидно, что такие расписания будут неприемлемыми, но такой подход способен охватить сразу всю область поиска уже на начальном этапе работы алгоритма. Модель начальной популяции решений представлена на рисунке 3.

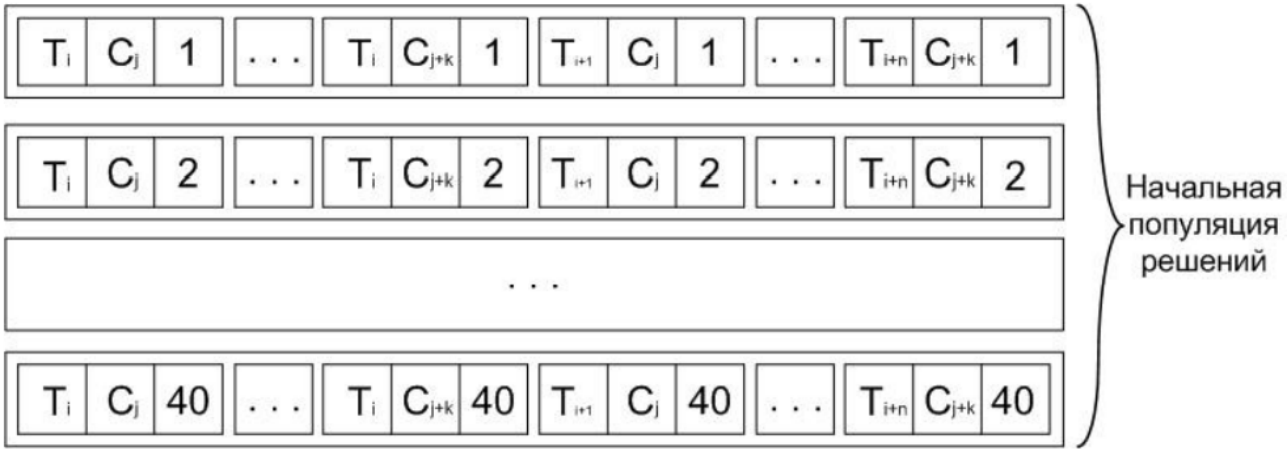


Рисунок 3 - Начальная популяция решений

Где $T_i$ – номер учителя, 0 ≤ i ≤ n, где n – количество учителей в школе – 1, $C_j$ – номер класса, 0 ≤ j ≤ k, где k – количество учителей в школе – 1.

Такая стратегия выбора начальной популяции чревата тем, что в процессе эволюции потенциально хорошие номера уроков могут исчезать, поэтому крайне важно хотя бы на ранних поколениях с некоторой периодичностью добавлять в текущую популяцию решений значения из начального поколения. Помимо этого в целях предотвращения сходимости алгоритма к локальному экстремуму имеет смысл со временем изменять значение вероятности возникновения мутации, которая представляет собой случайную замену номеров уроков всех занятий, проводимых каким-либо учителем.

В процессе формирования модели алгоритма следует учитывать и тот факт, что нарушение определенных правил недопустимо ни в коем случае. Такие правила называются жесткими ограничениями. В контексте рассматриваемой задачи это могут быть такие правила как: отсутствие разрывов между занятиями для учащихся, отсутствие коллизий в расписании и др. А пожелания участников учебного процесса, например, отсутствие разрывов между занятиями у учителей, можно считать мягкими ограничениями. Алгоритм будет стремиться выполнить эти требования и минимизировать число их нарушений, тем не менее, их наличие не будет оцениваться алгоритмом как недопустимое решение.

Существует несколько стратегий оценки приспособленности решений [5], например:

- Найти такое представление хромосомы, которое полностью или частично исключало бы саму возможность нарушения ограничений. Это существенно бы упростило решение задачи, но в рамках задачи по составлению расписания занятий такого представления нет.
- В процессе оценки решений отбрасывать нарушающие хотя бы одно жесткое ограничение. Такой подход может привести к возникновению ситуации, при которой ценная информация, содержащаяся в этих решениях, может быть навсегда утеряна.
- В процессе оценки решений исправлять те, которые нарушают хотя бы одно жесткое ограничение. Такой подход также может привести к потере ценной информации.
- В процессе оценки решений штрафовать те решения, которые нарушают жесткие ограничения. Такой подход меняет жесткое ограничение на мягкое, но с более высоким штрафом. При этом он лишь делает оценку плохих решений хуже, не

исключая полностью их из рассмотрения. Получается, что такие решения становятся менее желательными, но, тем не менее, информация, хранящаяся в них, не теряется. Единственной сложностью такого подхода является определение модели назначения штрафов, поскольку неправильно подобранный штраф может привести к тому, что недопустимое решение либо окажется оптимальным, либо полностью исключится из рассмотрения, то очень важно подобрать правильные значения штрафов за каждое нарушение жестких ограничений.

Так как рассматриваемая задача имеет большое количество параметров, критериев и условий, то решение задачи расположено в огромном пространстве потенциальных решений и, следовательно, необходимо использовать максимально широкую область поиска. Потеря ценной информации будет существенна и только сузит эту область. Поэтому целесообразно при решении задачи выбрать последний подход.

Чрезвычайно важно не допускать излишнюю потерю потенциально хороших генов во время эволюции и стараться давать возможность особям с низкой приспособленностью проходить отбор и, скрещиваясь с другими индивидуумами, в том числе и с наиболее приспособленными, передавать свою генетическую информацию новым поколениям. Для выполнения данной цели как раз подойдут такие методы отбора как: случайный, турнирный отбор, отбор по правилу рулетки, стохастическая универсальная выборка и ранжированный отбор. Эти виды селекции как раз подразумевают подобную особенность эволюции.

Хотя перечисленные методы селекции позволяют сохранить наличие некоторого количества потенциально хороших генов в процессе эволюции, в любой момент эволюции может произойти такая ситуация, при которой лучшие решения в текущем поколении не пройдут отбор и исчезнут. Потеря, скорее всего, будет временная, но возникновение таких потерь может увеличить время поиска лучшего решения может, а гарантия «возвращения» утерянных решений, на самом деле, отсутствует. Для того чтобы избежать подобных ситуаций, модель алгоритма можно реализовать с использованием стратегии элитизма. Суть данной стратегии заключается в том, что какое-то небольшое, заранее заданное число копий лучших (элитных) индивидуумов обязательно, до этапов отбора, скрещивания и мутации, переходит в будущее поколение [7]. Схема эволюционного цикла алгоритма с применением стратегии элитизма представлена на рисунке 4.

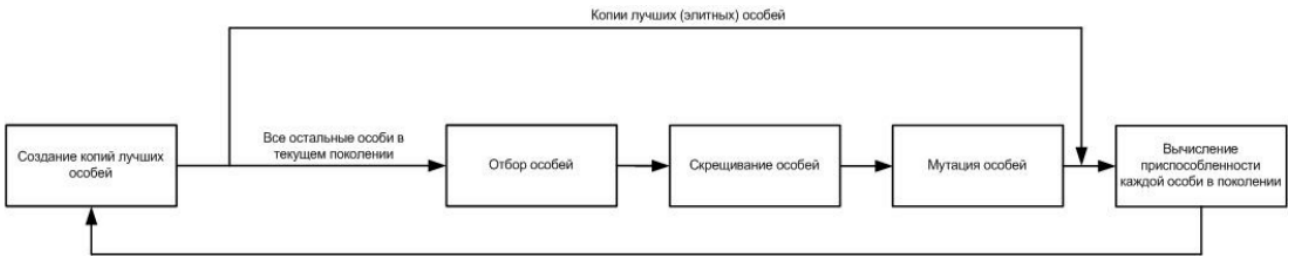


Рисунок 4 - Эволюционный цикл алгоритма с применением стратегии элитизма

В качестве модели скрещивания для выбранной модели хромосомы можно применить следующие методы скрещивания: одноточечное, двухточечное и равномерное скрещивание [6]. Перечисленные виды рекомбинации не нарушают структуру хромосомы и не создают

недопустимые гены в процессе своей работы. При этом они способны поддерживать многообразие комбинаций генов для всей популяции решений.

На рисунке 5 представлена зависимость максимальной и средней приспособленности от поколения для выбранной модели алгоритма с турнирным методом отбора и размером турнира 2, и равномерным скрещиванием. Красной линией представлена зависимость максимальной приспособленности, зелёной – зависимость средней. Модель генетического алгоритма была программно реализована с применением языка программирования Python и библиотеки Deap.

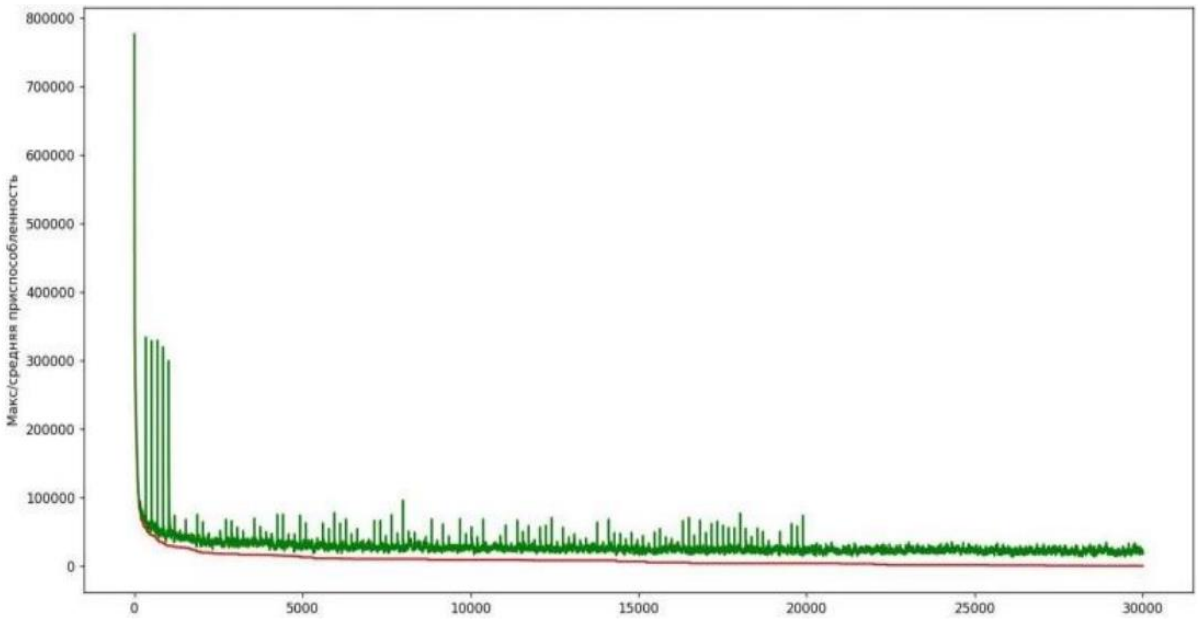


Рисунок 5 - График эволюции генетического алгоритма

Из графика видно, что алгоритм достаточно быстро преобразует начальное поколение, состоящее из решений, насчитывающих огромное количество коллизий, в решения с различными комбинациями номеров уроков. Но такому быстрому спуску графика сопутствует потеря различного числа номеров занятий для некоторых пар учитель-класс, вследствие чего поиск замедляется. На графике заметны резкие скачки зеленой линии – это участки эволюции, на которых проводилась процедура добавления в популяцию генотипов из начального поколения, а также промежутки увеличения значения графика средней приспособленности - повышение вероятности возникновения мутации для поддержания многообразия генов. Но буквально через несколько поколений после таких операций средняя приспособленность решений возвращалась к своему прежнему уровню, а максимальная улучшалась. Это свидетельствует о том, что выбранная стратегия не ухудшает популяцию, а вносит в нее утерянные в процессе эволюции комбинации генов, тем самым расширяя область поиска. Алгоритм, преодолевая преграды, возникшие на его пути, борется со стагнацией прогресса, поддерживает многообразие популяции потенциальных решений, избегает преждевременной сходимости к локальному экстремуму и находит решение за приемлемое время.

Разработанная коллективом авторов модель генетического алгоритма позволяет решать задачу составления расписания занятий в школе. Применение такого алгоритма для решения подобных задач позволит существенно сократить загруженность организаторов учебного

процесса и оптимизировать их рабочее время, а учебный процесс, построенный на основе такого расписания, будет протекать непрерывно и с оптимальными нагрузками на учащихся. Однако при формировании модели генетического алгоритма следует учитывать тот факт, что применение генетических алгоритмов на практике влечет за собой большой объем счетных операций и вероятность преждевременной сходимости к локальному экстремуму.

Данная модель может быть оптимизирована для различных учебных заведений путем добавления в представление хромосомы различных компонент, таких как наименование дисциплин и номера учебных аудиторий и других, в случае, если их влияние на составление расписание существенно.

**Список литературы**

1. Танаев, В.С. Введение в теорию расписаний / В.В., Шкурба – М.: Наука, 1975. – 256с.
2. Танаев, В.С. Теория Расписаний. Многостадийные системы / Ю.Н., Сотсков, В.А., Струсевич – М.: Наука, 1989. – 327с.
3. Вирсански, Э. Генетические алгоритмы на Python – М.: ДМК Пресс, 2020. – 286 с.
4. Джоши, П. Искусственный интеллект с примерами на Python / – СПб.: ООО «Диалектика», 2019. – 448с.
5. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / М. Пилиньский, Л. Рутковский. – М.: Горячая линия - Телеком, 2006. – 452 с.
6. Т.В., Панченко Генетические алгоритмы / под ред. Ю. Ю., Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с.
7. Amita Kapoor. Hands-On Artificial Intelligence for IoT. – Packt Publishing Ltd., 2019. – 370с.
8. Deap Documentation [Электронный ресурс] / Сайт с документацией по библиотеке Deap. URL: https://deap.readthedocs.io/en/master/index.html

**References**

1. Tanaev, V.S. Introduction to scheduling theory / V.V., Shkurba – M.: Science, 1975. – 256p.
2. Tanaev, V.S Schedule Theory. Multistage systems / Y.N., Sotskov, V.A., Strusevich - M.: Science, 1989. - 327p.
3. Wirsansky, E. Hands-On Genetic Algorithms with Python – M.: DMK Press, 2020. – 286 p.
4. Joshi, P. Artificial intelligence with examples in Python / - St. Petersburg: Dialectika, 2019. - 448 p.
5. Rutkovskaya, D. Neural networks, genetic algorithms and fuzzy systems / M. Pilinsky, L. Rutkovsky. - M.: Hot line - Telecom, 2006. - 452 p.
6. T.V., Panchenko Genetic algorithms / ed. Y. Y., Tarasevich. - Astrakhan: Astrakhan University Publishing House, 2007. - 87 p.
7. Amita Kapoor. Hands-On Artificial Intelligence for IoT. – Packt Publishing Ltd., 2019. – 370p.
8. Deap Documentation [Electronic resource] / Deap library documentation site. URL: https://deap.readthedocs.io/en/master/index.html