Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Воронежский государственный технический университет»

На правах рукописи

Баранов Дмитрий Алексеевич

ИНТЕЛЛЕКТУАЛИЗАЦИЯ СИСТЕМЫ ЦЕЛОЧИСЛЕННОЙ УСЛОВНОЙ ОПТИМИЗАЦИИ С ВАРИАТИВНЫМ ИСПОЛЬЗОВАНИЕМ ЭВОЛЮЦИОННЫХ АЛГОРИТМОВ

Специальность 2.3.1. Системный анализ, управление и обработка информации, статистика

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
Барабанов Владимир Федорович

Воронеж – 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы. Современные интеллектуальные системы находят широкое применение в различных сферах науки и техники, способствуя автоматизации сложных процессов, обработке данных и принятию решений в условиях неопределенности. Однако реальные задачи в таких направлениях, как оптимизация логистических маршрутов, конечных автоматов и игровых стратегий имеют множество критериев оценки и разнообразных ограничений, поэтому требуют использования эффективных методов оптимизации и интеллектуального поиска решений. В этом контексте эволюционные алгоритмы зарекомендовали себя как мощный инструмент для решения задач большой размерности с существенными вычислительными затратами. Существенный вклад в развитие этих методов внесли такие ученые, как Золотарюк А.В., Кажанов А.А., Курейчик В.М., Ногин В.Д., Пересветов В.В., Подвальный С.Л., Подиновский В.В., Штовба С.Д., D. Саймон, M. Dorigo, D. Karaboga.

Одной из актуальных задач в области применения эволюционных и интеллектуальных подходов является адаптация алгоритмов к решению задач целочисленной условной оптимизации. Традиционные детерминированные методы зачастую оказываются неэффективными из-за своей ограниченной гибкости и высокой вычислительной сложности, тогда как эвристические методы требуют тщательной настройки параметров и адаптации к конкретным условиям. Это обусловливает потребность в разработке интеллектуальных систем, обеспечивающих вариативность выбора алгоритмов и их адаптацию к специфике и ограничениям поставленных задач.

Эволюционные алгоритмы часто применяются изолированно, без учёта потенциальной синергии при их комбинировании. Интеллектуализация управления вариативными алгоритмами позволяет реализовать динамическое переключение стратегий оптимизации в зависимости от текущих характеристик задачи, что способствует повышению качества решений в задачах целочисленной условной оптимизации и повышает устойчивость к застреванию в локальных субоптимальных решениях.

Дополнительную значимость исследованию придает широкая область возможного применения разрабатываемых решений – от транспортных задач и анализа поведения конечных автоматов до моделирования и планирования сложных процессов. Это позволяет использовать предложенную методологию как в рамках теоретических исследований, так и для решения прикладных задач в различных отраслях.

Таким образом, актуальность темы диссертационного исследования обусловлена необходимостью разработки вычислительных и управляющих систем, в основе которых лежит использование методов искусственного интеллекта, позволяющих организовать процесс оптимизации как комплексный цикл анализа данных и принятия решений.

Тематика диссертационной работы соответствует научному направлению ФГБОУ ВО «Воронежский государственный технический университет» «Вычислительные комплексы и проблемно-ориентированные системы управления».

Целью работы является повышение эффективности решения задач целочисленной условной оптимизации за счет создания интеллектуальной системы, предусматривающей вариативное применение эволюционных алгоритмов.

Задачи исследования. Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести системный анализ эволюционных алгоритмов и определить направления их модификации для решения задач целочисленной условной оптимизации;
2. Разработать математическое обеспечение задачи целочисленной условной оптимизации с учетом вида ограничений и возможностью задания гибких правил и условий задачи;
3. Разработать специальную структуру алгоритмического обеспечения систем целочисленной условной оптимизации, включающую

верификацию промежуточных решений эволюционных алгоритмов и механизм штрафов по критериям для повышения качество решений;

4. Модифицировать эволюционные алгоритмы для решения задач целочисленной условной оптимизации и интегрировать в них механизм верификации решений;

5. Реализовать модель интеллектуального выбора конфигураций эволюционных алгоритмов, позволяющей учитывать характеристики задачи для повышения эффективности вычислений;

6. Разработать интеллектуальную систему целочисленной условной оптимизации с вариативным использованием эволюционных алгоритмов, провести ее апробацию и оценить эффективности по показателям производительности и качества решений.

Объект исследования: интеллектуальные вычислительные и управляющие системы, предназначенные для решения задач целочисленной условной оптимизации.

Предмет исследования: Методы и алгоритмы эволюционного поиска, модели верификации решения и интеллектуального выбора стратегий, направленные на повышение эффективности решения задач целочисленной условной оптимизации.

Методы исследования: в ходе работы над диссертационным исследованием использовались методы системного анализа, принятия решений, эволюционных вычислений, машинного обучения и программной инженерии.

Тематика работы соответствует следующим пунктам паспорта специальности 2.3.1 «Системный анализ, управление и обработки информации, статистика»: п. 2 Формализация и постановка задач системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта; п. 4 Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений, обработки информации и искусственного интеллекта; п. 5 Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации,

управления, принятия решений, обработки информации и искусственного интеллекта; п. 10 Методы и алгоритмы интеллектуальной поддержки при принятии управленческих решений в технических системах;

Научная новизна работы: В диссертации получены следующие результаты, характеризующиеся научной новизной:

1. Математическое обеспечение задачи целочисленной условной оптимизации, отличающееся гибкой интеграцией набора эволюционных алгоритмов и возможностью их конфигурации для решения задач с различными типами ограничений, что повышает качество оптимизации.

2. Модель интеграции штрафных функций в эволюционные алгоритмы для задач целочисленной условной оптимизации, отличающаяся селективным применением штрафов по каждому критерию оптимизации в зависимости от типа нарушенного ограничения, что обеспечивает направленную коррекцию поиска решений и повышает скорость сходимости к допустимой области.

3. Алгоритмическое обеспечение систем целочисленной условной оптимизации, отличающееся наличием механизма верификации промежуточных решений с применением штрафов и многокритериальной оценки, что обеспечивает раннее выявление и сокращение количества недопустимых решений.

4. Интеллектуальная модель вариативного выбора конфигураций эволюционных алгоритмов на основе формализованного представления ограничений задачи, отличающаяся использованием трансформерной архитектуры для анализа структуры ограничений, формирования начальной стратегии поиска и повышения эффективности оптимизации.

5. Структура адаптивной интеллектуальной системы целочисленной условной оптимизации, включающая библиотеки модифицированных эволюционных алгоритмов, отличающаяся применением интеллектуальной модели и базы знаний для формирования стратегии решения задачи, что обеспечивает устойчивость и предсказуемость процесса нахождения решений.

Теоретическая и практическая значимость исследования заключается в разработке комплекса теоретических и прикладных решений, направленных на повышение эффективности применения эволюционных алгоритмов при решении задач целочисленной условной оптимизации. В рамках исследования предложены: формальные средства описания ограничений с поддержкой вложенных логических выражений, математическая модель верификации решений, а также интеллектуальная система вариативного использования эволюционных алгоритмов, использующая трансформерную архитектуру для формального анализа структуры ограничений и адаптивного выбора вычислительной стратегии.

Теоретическая значимость заключается в развитии методов системного анализа и принятия решений, формализации ограничений и интеграции адаптивных эволюционных стратегий, что способствует расширению научных представлений об эффективных подходах к решению сложных задач целочисленной условной оптимизации.

Практическая значимость состоит в возможности применения разработанного инструментария в вычислительных системах, предназначенных для оптимизации в условиях реальных ограничений и многокритериальности, в том числе в транспортных задачах, моделировании процессов, управлении техническими объектами и других прикладных областях. Полученные результаты могут быть использованы в научно-исследовательских и проектных организациях, занимающихся разработкой интеллектуальных систем поддержки принятия решений и оптимизации.

Положения, выносимые на защиту

1. Математическое обеспечение задачи целочисленной условной оптимизации, основанное на вариативном использовании эволюционных алгоритмов и возможности их конфигурации для решения задач с различными типами ограничений, что обеспечивает повышение качества оптимизации.

2. Модель интеграции штрафных функций в эволюционные алгоритмы для задач целочисленной условной оптимизации обеспечивает

селективное применение штрафов по каждому критерию оптимизации в зависимости от типа нарушенного ограничения, что позволяет осуществлять направленную коррекцию поиска решений и повышает скорость сходимости допустимой области.

3. Алгоритмическое обеспечение систем целочисленной условной оптимизации верифицирует промежуточные решения эволюционных алгоритмов с использованием штрафных функций и разработкой критериев и моделей оценки эффективности, что обеспечивает раннее выявление и сокращение количества недопустимых решений.

4. Модель интеллектуального выбора конфигураций эволюционных алгоритмов на основе формализованных ограничений задачи обеспечивает формирование начальной стратегии поиска и повышение эффективности оптимизационного процесса.

5. Структура адаптивной интеллектуальной системы целочисленной условной оптимизации содержит библиотеки модифицированных эволюционных алгоритмов, использует интеллектуальную модель и базу знаний для их переключения, обеспечивая устойчивость и предсказуемость процесса нахождения решений.

Результаты внедрения. Разработанные материалы внедрены в образовательный и научно-исследовательский процесс кафедры автоматизированных и вычислительных систем Воронежского государственного технического университета и в компаниях «Стартап», «Бренд 42», «Сател ПрО».

Апробация работы. Основные положения диссертационной работы докладывались и обсуждались в ряде конференций, среди которых: Оптимизация и моделирование в автоматизированных системах (Воронежский государственный технический университет, 2023), Нано-био-технологии, теплоэнергетика, математическое моделирование (Липецкий государственный технический университет, 2024), «Антропоцентрические науки в образовании: вызовы, трансформации, ресурсы» (Воронежский государственный технический университет, 2024), Цифровые системы и модели: теория и практика

проектирования, разработки и использования (Казанский государственный энергетический университет, 2025), Интеллектуальные технологии цифровой инженерии (Воронежский государственный технический университет, 2025), Международная молодежная научная школа «Оптимизация и моделирование в автоматизированных системах» (Воронежский государственный технический университет, 2025).

Достоверность результатов обусловлена корректным использованием теоретических методов исследования и подтверждена результатами проведенных вычислительных экспериментов и сравнительным анализом данных.

Публикации. По результатам диссертационного исследования опубликовано 15 научных работ (6 – без соавторов), в том числе 6 – в изданиях, рекомендованных ВАК РФ, 3 свидетельства о регистрации программы для ЭВМ. В работах, опубликованных в соавторстве и приведенных в конце реферата, автором получены следующие результаты: [31, 32, 38] – алгоритмическое обеспечение муравьиного алгоритма, [41] – алгоритмическое обеспечение пчелиного алгоритма, [44] – алгоритмическое обеспечение алгоритма имитации отжига, [48] – описание средств формализации представления ограничений, [56] – структура системы, [82, 84, 85, 87] – отбор конфигураций алгоритмов для интеллектуального выбора стратегии решения, [88] – интеллектуализация системы дискретной (целочисленной условной) оптимизации.

Структура и объем работы: Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы из 102 наименований, 2 приложений. Основная часть изложена на 164 страницах с 51 рисунком и 21 таблицей.

ГЛАВА 1. ПРОБЛЕМАТИКА ЦЕЛОЧИСЛЕННОЙ УСЛОВНОЙ ОПТИМИЗАЦИИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

1.1 Проблематика целочисленной условной оптимизации

Целочисленная условная оптимизация (ЦУО) представляет собой раздел математического программирования [1], ориентированный на задачи, в которых целевая функция подлежит минимизации или максимизации при наличии системы ограничений, а переменные могут принимать только целочисленные значения. Такие модели занимают промежуточное положение между непрерывной и дискретной оптимизацией: с одной стороны, они оперируют целочисленной, дискретной природой решений, с другой – опираются на формальные средства классического математического программирования, включая линейные и нелинейные ограничения.

ЦУО широко применяется в задачах ресурсного планирования, маршрутизации, составления расписаний, проектирования технических систем и операционного управления [2]. Практический смысл обусловлен тем, что многие реальные объекты – количество автомобилей, сотрудников, контейнеров, шагов маршрута, единиц продукции – являются именно целочисленными. Ограничения отражают физические, технологические, логистические, временные, структурные и иные условия, которые решение обязано удовлетворять.

Допустимое множество представляет собой комбинацию целочисленных наборов, удовлетворяющих ограничениям. В отличие от непрерывных задач, пространство решений «распадается» на отдельные точки, не образующие гладкую поверхность. Это исключает возможность использования градиентных методов и существенно усложняет процесс поиска.

Ограничения в ЦУО могут быть:

- линейными и нелинейными;
- равенствами или неравенствами;
- логическими, структурными, комбинаторными;
- жесткими и мягкими.

Именно структура и тип ограничений чаще всего определяют вычислительную сложность задачи. Даже при небольшом числе переменных количество допустимых комбинаций растет экспоненциально [3]. Это явление, известное как «комбинаторный взрыв», приводит к тому, что большинство задач ЦУО относится к NP-трудным. Полный перебор вариантов практически невозможен, что определяет необходимость специализированных методов.

В разных моделях встречаются следующие типы задач: целочисленные задачи, бинарные (булевые), смешанные целочисленно-непрерывные модели [4]. Последние особенно востребованы т.к. позволяют сочетать логические решения с непрерывными параметрами ресурсов и потоков.

Далее рассмотрены преимущества применения ЦУО [5].

Целочисленность напрямую отражает целочисленную природу многих объектов и действий. Поэтому решения обладают физической интерпретируемостью (например, нельзя распределить 2,7 автомобиля или отправить 0,4 сотрудника в смену).

Через формулировку ЦУО можно выразить такие задачи, как задачу раскроя, упаковки, связанные с графовыми структурами (построение маршрутов), календарного и логистического планирования, оптимального распределения ресурсов, конфигурационные и структурные задачи [6].

Для ЦУО разработаны точные методы (ветвей и границ, отсечения, метод Бендера и др), приближенные (релаксации, декомпозиции) и эвристические (жадные алгоритмы, локальный поиск, эволюционные методы и пр.) [7, 8, 9]. Комбинация методов позволяет получать приемлемые решения даже при высокой размерности.

Наряду с этим, ЦУО сталкивается с следующими трудностями и ограничениями:

– экспоненциальная вычислительная сложность. Большинство задач ЦУО относится к NP-полному или NP-трудному классам [10]. Рост числа переменных или ограничений немедленно приводит к кратному росту сложности;

– ограниченность точных методов. Методы полного перебора, динамического программирования или ветвей и границ применимы лишь к задачам малой и средней размерности. При больших моделях требования по времени и памяти становится чрезмерными;

– чувствительность к структуре ограничений. Даже незначительное изменение модели (добавления логического ограничения или нового типа связей) может радикально усложнить решение;

– сложность формализации задачи. Для корректной постановки требуется точно определить переменные, формализовать все ограничения и выбрать адекватную целевую функцию. Ошибки на этапе моделирования нередко приводят к некорректным решениям.

Таким образом, несмотря на развитость математического аппарата ЦУО, существующие методы демонстрируют ограниченную масштабируемость и высокую чувствительность к структуре ограничений. Это приводит к необходимости разработки адаптивных и интеллектуальных подходов, способных учитывать специфику ограничений и динамически подбирать стратегию оптимизации.

1.2 Современные тенденции применения искусственного интеллекта в эволюционных алгоритмах

Современные исследования в области эволюционных алгоритмов демонстрируют устойчивую тенденцию к их интеграции с методами искусственного интеллекта (далее – ИИ), направленной на повышение эффективности решения задач целочисленной условной оптимизации. В рамках данного направления ИИ используется не только для ускорения вычислений, но и для интеллектуализации процессов управления поиском, адаптации алгоритмов и анализа структуры решаемых задач.

Одним из традиционных подходов является использование ИИ-моделей для аппроксимации значений целевой функции [11] или приспособленности решений, что позволяет сократить вычислительные затраты в задачах,

требующих сложного моделирования. Подобные методы нашли применение в задачах аэродинамики, биоинформатики и других областях, где прямое вычисление функции качества является ресурсоемким. Для этих целей применяются различные модели, включая ансамблевые методы и нейронные сети, обучаемые на данных, получаемых в ходе первых итераций эволюционного поиска.

Наряду с этим, все более распространенным становится применение ИИ для адаптивного управления параметрами эволюционных алгоритмов [12]. Вместо использования фиксированных значений вероятностей мутации, кроссовера и других операторов, интеллектуальные модели анализируют динамику процесса оптимизации и формируют рекомендации по их корректировке в реальном времени, что повышает устойчивость алгоритмов и снижает зависимость от ручной настройки [13].

Активно развивающейся тенденцией является использование ИИ для предсказания стратегии решения задачи, а не для оценки отдельных кандидатов [14]. В рамках данного подхода интеллектуальная модель анализирует формализованное описание задачи, включая систему ограничений, и на этой основе определяет целесообразный сценарий применения эволюционных алгоритмов: выбор алгоритма, режим его работы, параметры операторов и возможные стратегии переключения между методами. Такой подход особенно эффективен в задачах ЦУО, где именно ограничения в значительной степени определяют сложность допустимого пространства решений.

Развитие глубоких нейронных архитектур, в том числе моделей трансформерного типа, способствует расширению возможностей анализа сложных логических и структурированных описаний задач. Механизмы внимания позволяют учитывать взаимосвязи между ограничениями, выявлять критические условия и формировать стратегические рекомендации по организации эволюционного поиска без прямого вмешательства в процесс генерации решений.

Дополнительным направлением является применение интеллектуальных методов в многокритериальных задачах оптимизации, где требуется учитывать несколько противоречивых целей [15]. ИИ используется для анализа

компромиссов между критериями, аппроксимации Парето-фронта и поддержки принятия решений при выборе предпочтительных решений [16, 17]. В динамических задачах интеллектуальные модели также применяются для адаптации поведения ЭА к изменяющимся условиям и ограничениям [18].

Несмотря на значительный прогресс, интеграция ИИ в эволюционные алгоритмы сопровождается рядом проблем, среди которых выделяются высокая вычислительная сложность, ограниченная интерпретируемость гибридных систем и необходимость соблюдения баланса между глобальным эволюционным поиском и локальной адаптацией. Решение этих проблем остается предметом активных исследований.

В целом, современные тенденции применения искусственного интеллекта в эволюционных алгоритмах характеризуются смещением акцента от локального ускорения отдельных этапов вычислений к интеллектуальному управлению стратегией оптимизации в целом. Такой подход расширяет области применимости эволюционных алгоритмов и формирует основу для создания адаптивных вычислительных комплексов, способных эффективно решать широкий класс задач целочисленной условной оптимизации.

1.3 Применение ограничений к задачам целочисленной условной оптимизации

Задачи ЦУО в большинстве практических приложений характеризуются наличием не только одной, но нескольких, зачастую противоречивых целей. Например, в задачах маршрутизации требуется одновременно минимизировать длину маршрута, время выполнения, стоимость и риск, при этом улучшение одного критерия нередко приводит к ухудшению других. Такая постановка приводит к многокритериальным моделям с ограничениями, в которых допустимая область решений формируется сложной системой условий [19].

С формальной точки зрения задача целочисленной условной оптимизации может быть представлена в виде выражения (1.1):

$$\min_{x \in \mathbb{Z}^n} f(x), \text{при } g_i(x) \leq 0,\ h_j(x) = 0, \quad (1.1)$$

где x — вектор целочисленных переменных, *f(x)* — целевая функция, $g_i(x)$ и $h_j(x)$ — функции ограничений. В отличие от задач безусловной оптимизации, наличие ограничений существенно сужает допустимую область и делает процесс поиска решения более сложным и чувствительным к структуре модели.

Ограничения в задачах ЦУО играют ключевую роль, поскольку именно они определяют допустимость решений и во многом формируют топологию пространства поиска. В реальных задачах ограничения могут моделировать временные окна, структурные зависимости между элементами решения, ограничения ресурсов, логические правила следования, пропускные способности, приоритеты и исключения. Таким образом, ограничения являются не второстепенным элементом модели, а основным фактором, определяющим сложность задачи.

С точки зрения обработки в эволюционных алгоритмах ограничения можно классифицировать следующим образом:

– жесткие ограничения, нарушение которых делает решение недопустимым и приводит к его исключению из дальнейшего рассмотрения;

– мягкие (штрафные) ограничения, допускающие нарушение с наложением штрафа на значение функции приспособленности;

– логические и структурные ограничения, задающие допустимые отношения между элементами решения (порядок следования, принадлежность, взаимное исключение);

– ресурсные и временные ограничения, ограничивающие суммарные характеристики решения.

На практике наиболее распространенным подходом к учету ограничений в эволюционных алгоритмах является использование штрафных функций, при которых модифицированная функция приспособленности представляется в соответствии с выражением (1.2):

$$f(x) = f(x) + \sum_i \lambda_i \times \max(0, g_i(x)), \quad (1.2)$$

где λ_i — коэффициенты штрафа. Однако эффективность данного подхода существенно зависит от выбора параметров штрафа и структуры ограничений. При некорректной настройке штрафные функции либо не обеспечивают соблюдение ограничений, либо чрезмерно подавляют процесс оптимизации.

Альтернативным подходом является полное исключение недопустимых решений из популяции. Такой метод обеспечивает строгую корректность, но при высокой плотности ограничений может привести к резкому сокращению допустимой области и деградации эволюционного поиска. В задачах с логическими зависимостями и комбинированными правилами (например, условия следования, исключения или обязательного присутствия элементов) данный эффект проявляется особенно сильно.

Особую сложность представляют задачи, в которых ограничения не являются независимыми, а образуют сложную систему взаимосвязей. В таких случаях даже незначительное изменение одного ограничения может радикально изменить структуру допустимого множества, что делает традиционные универсальные алгоритмы малоэффективными. Это обстоятельство особенно характерно для задач маршрутизации, календарного планирования и конфигурационной оптимизации [20].

Таким образом, ограничения в задачах целочисленной условной оптимизации не только усложняют процесс поиска решения, но и выступают основным источником информации о характере задачи. Эффективное управление ограничениями требует не только корректного выбора метода их учета, но и адаптации стратегии оптимизации к их структуре и жесткости. Это делает актуальной разработку интеллектуальных подходов, ориентированных на анализ и интерпретацию системы ограничений как ключевого элемента постановки задачи.

1.4 Системный анализ эволюционных алгоритмов

Для дальнейшей работы избрано 4 эволюционных алгоритма:

– Генетический алгоритм (ГА) [21, 22];

– Алгоритм муравьиной колонии (АМК) [23, 24, 25];

– Алгоритм пчелиной колонии (АПК) [26];

– Алгоритм имитации отжига (АИО) [27, 28]. Хотя данный алгоритм формально не относится к популяционным эволюционным методам, в рамках данной работы он рассматривается как метаэвристический компонент гибридной системы оптимизации.

Характеристики алгоритмов приведены на таблице 1.1.

Таблица 1.1 – Характеристика выбранных эволюционных алгоритмов

Характеристика	Генетический алгоритм	Алгоритм муравьиной колонии	Алгоритм пчелиной колонии	Алгоритм имитации отжига
Основная идея	Эволюция популяции решений через кроссовер и мутацию	Имитация поиска кратчайших путей муравьями с использованием феромонов	Моделирование поведения пчел-разведчиков и сборщиков	Термодинамическая релаксация с управляемым «охлаждением»
Тип задач	Дискретные и непрерывные (оптимизация гиперпараметров, задача коммивояжера и т.д.)	Дискретные (графы, маршрутизация)	Непрерывные (функции с множеством экстремумов)	Унимодальные и мультимодальные (выход из локальных оптимумов)
Преимущества	Универсальность, параллельность	Эффективность для маршрутных задач, самоорганизация через феромоны	Быстрый поиск глобальных оптимумов в сложных пространствах	Простота реализации, устойчивость к локальным минимумам
Недостатки	Высокая вычислительная сложность, риск преждевременной сходимости	Зависимость от настройки параметров	Требует тонкой настройки для задач ЦУО	Относительно медленная сходимость
Причина выбора	Глобальный поиск, гибкость кодирования	Специализация на маршрутных задачах, учет динамических факторов через феромоны	Альтернатива ГА для непрерывных критериев	Локальная «шлифовка» решений

Каждый из выбранных алгоритмов имеет свои сильные стороны:

– генетический алгоритм основан на механизмах наследования, селекции и мутации. Он обеспечивает широкий охват пространства решений и особенно эффективен в целочисленных задачах, где необходимо сбалансировать противоречивые цели (например, минимизация расстояния маршрута при одновременном учёте времени и риска).

– алгоритм муравьиной колонии моделирует коллективное поведение насекомых при построении кратчайших путей. Он хорошо зарекомендовал себя в задачах маршрутизации и графовых структурах, где феромонные следы позволяют эффективно распределять поиск.

– алгоритм пчелиной колонии имитирует работу разведывательных и рабочих пчел, обеспечивая быструю локальную оптимизацию. Он часто используется как «доводчик» решения после глобального поиска, позволяя уточнить найденный результат.

– алгоритм имитации отжига основан на аналогии с процессом кристаллизации металла. Он способен преодолевать локальные минимумы и надёжно находить хорошие решения на финальной стадии оптимизации, хотя требует большего времени.

В интеллектуальных системах целесообразно не ограничиваться одним алгоритмом, а комбинировать их. Для этого используется контроллер, который анализирует текущее состояние поиска и переключает алгоритмы в зависимости от ситуации. Например, он может активировать АМК при падении разнообразия популяции в ГА, запустить АПК для ускоренной локальной оптимизации или применить АИО, когда система «застряла» в локальном оптимуме. Такая интеграция позволяет минимизировать слабые стороны отдельных методов и объединить их сильные стороны, что особенно важно в условиях высокой размерности и динамики реальных задач.

1.5 Формализация эволюционных алгоритмов

1.5.1 Генетический алгоритм: эволюция через наследственность

Генетический алгоритм (ГА) — это классический представитель эволюционных алгоритмов, впервые предложенный Джоном Холландом в 1970-х годах. Он основывается на идеях естественного отбора и генетики, описанных Чарльзом Дарвином, и имитирует процесс эволюции популяции особей, где каждая особь представляет собой потенциальное решение задачи [21, 22]. ГА особенно эффективен для задач с большим числом переменных, целочисленным или непрерывным пространством поиска [23].

Для данного алгоритма свойственен ряд важных операторов, которые также представляют собой основные этапы работы:

– инициализация популяции: создается начальная популяция особей, где каждая особь представлена в виде хромосомы (например, бинарной строки, вещественного вектора или другой структуры данных). Хромосома кодирует решение задачи;

– оценка приспособленности: для каждой особи вычисляется значение целевой функции (функции приспособленности), которая определяет, насколько хорошо данное решение справляется с задачей;

– отбор: выбираются особи, которые будут участвовать в создании следующего поколения. Часто используется метод рулеточного отбора, турнирного отбора или элитизма (сохранение лучших особей без изменений);

– скрещивание: выбранные особи (родители) обмениваются частями своих хромосом для создания потомков. обмен генами между особями. Например, если существуют особи $A = \{a_1, a_2, a_3, a_4\}$ и $B = \{b_1, b_2, b_3, b_4\}$, то в результате одноточечного скрещивания образуются две дочерние особи $C = \{a_1, a_2, b_3, b_4\}$ и $D = \{b_1, b_2, a_3, a_4\}$. Имеется множество других видов скрещивания: двухточечное, равномерное и т.д.;

– мутация: с небольшой вероятностью в хромосоме изменяются отдельные гены. Для вещественных векторов мутация может быть реализована через добавление случайного шума.

Генетические алгоритмы широко используются в задачах оптимизации маршрутов, планирования, машинного обучения (например, для подбора гиперпараметров моделей), а также в инженерном проектировании (оптимизация конструкций).

1.5.2 Алгоритм муравьиной колонии: коллективный интеллект колоний

Алгоритм муравьиной колонии был предложен Марко Дориго в 1992 году и основывается на поведении муравьев, которые находят кратчайшие пути между гнездом и источником пищи благодаря феромонным следам. Этот алгоритм особенно эффективен для задач целочисленной оптимизации, таких как задача коммивояжера, задачи планирования маршрутов и сетевой оптимизации [24, 25, 26].

Основой алгоритма является феромонный след, который:

– усиливается на оптимальных путях;

– испаряется со временем, предотвращая застревание в локальных оптимумах;

Ниже приведен основной механизм работы:

– инициализация: задается граф, на котором будут работать муравьи (например, города и пути между ними в задаче TSP). Каждое ребро графа инициализируется начальным уровнем феромона τ_{ij}, обычно небольшим положительным значением.

– Для каждого муравья и его шага вероятностный выбор пути в соответствии с формулой (1.1):

$$P_{ij} = \frac{(\tau_{ij})^{\alpha} \times (\eta_{ij})^{\beta}}{\sum_{k=1}^{l} (\tau_{ik})^{\alpha} \times (\eta_{ik})^{\beta}}, \quad (1.1)$$

где k –множество еще не посещенных городов;

η – сравнительный показатель целесообразности выбора пути (например, обратная длина пути);

α – параметр, отвечающий за склонность следования по феромонам;

β – параметр, отвечающий за склонность следования по короткому пути (степень «жадности»);

– обновление феромонов в соответствии с формулой (1.2)

$$\tau_{ij} = (1 - \rho) \times \tau_{ij} + \Delta\tau_{ij}, \quad (1.2)$$

где ρ – коэффициент испарения; $\Delta\tau_{ij}$ – сумма феромонов от всех муравьев, прошедших по ребру ij.

Как можно заметить, одной из особенностей алгоритма является требование тонкой настройки различных параметров, которые в свою очередь будут влиять на интенсивность поиска, чувствительность к локальным минимумам и скорость сходимости. Алгоритм муравьиной колонии широко применяется в раздачах маршрутизации (например, в телекоммуникационных сетях), планировании логистики и управления трафиком.

1.5.3 Алгоритм пчелиной колонии: разведка и эксплуатация ресурсов

Алгоритм пчелиной колонии был предложен Д. Карабогой в 2005 году [27] и моделирует поведение пчел при поиске пищи. Этот алгоритм сочетает в себе разведку и освоение пространства решений, что делает его подходящим для задач глобальной оптимизации, особенно в непрерывных пространствах.

В рамках его работы, особи разделяются на два вида:

– пчелы-разведчики: исследуют случайные области;

– пчелы-рабочие: углубляются в перспективные зоны.

Алгоритм пчелиной колонии состоит из следующих этапов:

– создается начальная популяция источников пищи (потенциальных решений), где каждый источник представлен вектором x_i. Половина пчел считается разведчиками, а другая половина — рабочими;

– фаза разведки: пчелы разведки случайным образом исследуют пространство решений и оценивают приспособленность каждого источника пищи по целевой функции;

– фаза освоения: пчелы-рабочие выбирают источники пищи на основе их качества (приспособленности) и выполняют локальный поиск вокруг выбранных источников. Новое решение генерируется по формуле (1.3)

$$v_{ij} = x_{ij} + \phi_{ij} \times \left(x_{ij} - x_{kj}\right), \quad (1.3)$$

где ϕ_{ij} – случайное число в диапазоне [-1;1]; x_{kj} – соседний источник пищи.

– оценка и обновление: если новое решение лучше старого, оно заменяет старое. Если источник пищи не улучшается после заданного числа попыток, он считается заброшенным, а разведчик будет искать новый источник;

Из преимуществ также можно выделить следующие:

– хороший баланс между разведкой и эксплуатацией;

– устойчивость к локальным экстремумам в непрерывных пространствах;

– простота реализации.

Алгоритм пчелиной колонии широко используется в задачах оптимизации параметров нейронных сетей, кластеризации данных и инженерного проектирования.

1.5.4 Алгоритм имитации отжига: термодинамическая релаксация

Метод имитации отжига был предложен Скоттом Киркпатриком в 1983 году и вдохновлен процессом отжига в металлургии, где металл нагревают и медленно охлаждают для уменьшения дефектов кристаллической решетки. Этот метод относится к метаэвристикам и используется для поиска глобального оптимума в сложных пространствах поиска [28, 29].

Принцип работы метода следующий:

– инициализация: задается начальное решение x и начальная температура T_0. Температура T играет роль параметра, контролирующего вероятность принятия худших решений.

– генерация нового решения: создается новое решение x' в окрестности текущего решения x с применением, например, мутационной функции;

– принятие решения: если $f(x') > f(x),\ x'$ принимается, иначе, x' принимается с вероятностью, вычисляемой по формуле (1.4). Это позволяет алгоритму избегать застревания в локальных минимумах

$$P = e^{\frac{f(x')-f(x)}{T}}; \tag{1.4}$$

– понижение температуры в соответствии с формулой (1.5)

$$T_{k+1} = \varsigma \times T_k, \text{где } \varsigma < 1, \tag{1.5}$$

где ς – коэффициент охлаждения.

Метод имитации отжига применяется в задачах размещения элементов на печатных платах, оптимизации маршрутов и планировании производственных процессов.

1.6 Интеллектуализация процесса решения задач целочисленной условной оптимизации

Интеллектуализация процесса решения задач ЦУО в разрабатываемой системе основана на принципе предварительного анализа ограничений, определяющих допустимую область. В отличие от существующих подходов, интеллектуальный модуль не анализирует структуру пространства решений напрямую, а использует представление ограничений как первичный источник информации.

На первом этапе интеллектуальный модуль интерпретирует предоставленный пользователем набор правил – как жестких, так и штрафных – и автоматически формирует перечень алгоритмов, корректно работающих с

заданной логикой ограничений. Методы, несовместимые с данным типом условий (например, не поддерживающие сложные логические зависимости между элементами решения или определенные схемы шифрования), исключаются до начала оптимизации. Это позволяет избежать применения заведомо нерелевантных алгоритмов и адаптировать процесс под конкретную постановку задачи.

После формирования допустимого перечня интеллектуальная модель выполняет классификацию ограничений и прогнозирует, какой из доступных алгоритмов обладает наибольшим потенциалом эффективности. В отличие от подходов, основанных на анализе структуры графов или характеристик целевых функций, предлагаемая схема опирается исключительно на свойства ограничений: их количество, типы, степень жесткости, наличие зависимостей, объем исключений и влияние на площадь допустимой области. Обучение модели проводится на выборке, содержащей различные системы ограничений и результаты работы методов оптимизации в соответствующих задачах. Это позволяет системе выявлять, какие сочетания ограничений приводят к успешной работе тех или иных алгоритмов.

Дальнейший процесс оптимизации также поддерживается модулем управления алгоритмов. Система контролирует изменения значения функции приспособленности и стабильность хода поиска. Если выбранный алгоритм перестает демонстрировать прогресс в рамках допустимой «терпимости», модуль выполняет выбор следующего метода, спрогнозированного ранее на основе ограничений. Такой подход предотвращает переключение на методы, не совместимые с логикой задачи, и сохраняет корректность поиска.

1.7 Цель работы и задачи исследования

Целью работы является повышение эффективности решения задач целочисленной условной оптимизации за счет разработки интеллектуальной системы.

Для достижения поставленной цели необходимо решить следующие задачи:

- провести системный анализ эволюционных алгоритмов и определить направления их адаптации и модификации для решения задач целочисленной условной оптимизации;
- выполнить математическую формализацию задачи целочисленной условной оптимизации и разработать средства описания ограничений, обеспечивающие задание гибких логических правил и условий задачи;
- разработать структуру алгоритмического обеспечения систем целочисленной условной оптимизации, включающую механизмы верификации промежуточных решений и учета ограничений в процессе эволюционного поиска;
- модифицировать эволюционные алгоритмы для решения задач целочисленной условной оптимизации с учетом разработанных механизмов верификации и санкционирования недопустимых решений;
- разработать модель интеллектуального выбора стратегии решения задачи целочисленной условной оптимизации, основанную на анализе системы ограничений, с целью повышения эффективности вычислений;
- провести апробацию модели интеллектуального выбора стратегии решения задачи целочисленной условной оптимизации и выполнить оценку эффективности по показателям производительности и качества получаемых решений.

ГЛАВА 2. РАЗРАБОТКА МАТЕМАТИЧЕСКОГО И АЛГОРИТМИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ ЦЕЛОЧИСЛЕННОЙ УСЛОВНОЙ ОПТИМИЗАЦИИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

2.1 Формализация задачи целочисленной условной оптимизации: модель, ограничения и цели

Пусть имеется задача размерностью n при $n \geq 2$, множество индексов элементов $N = \{0,1,2,\dots,n-1\}$ и П – множество всех перестановок длины n над N. Решение задачи представляет собой перестановку согласно выражению (2.1).

$$x = (x_1, x_2, \dots, x_n) \in \Pi; x_i \in N; x_i \neq x_j \quad \forall i \neq j \tag{2.1}$$

Для задачи определены критерии в количестве m и правила ограничений в количестве r. Для каждого критерия $k \in K$ определены:

– $A^{(k)} = [a_{ij}^{(k)}] \in R\hat{}(n \times n)$ – матрица стоимостей (метрик), где $a_{ij}^{(k)} \in R$ – стоимость перехода от элемента i к элементу j по критерию k;

– $d^{(k)} \in \{min, max\}$равление оптимизации;

– $c^{(k)} \in \{0,1\}$ – параметр цикличности, где $c^{(k)} = 1$ – циклический путь, $c^{(k)} = 0$ – линейный путь.

Условия допустимости могут уточняться множеством правил $R = \{r_1, r_2, \dots, r_n\}$, где каждое правило задается логическим предикатом, согласно уравнению (2.2)

$$\phi_r(x) = \begin{cases} 1, & \text{если правило нарушено} \\ 0, & \text{если правило выполняется} \end{cases} \tag{2.2}$$

Правила разделяются на жесткие $R^{\text{жест}}$ и мягкие $R^{\text{мягк}}$. Для жестких правил множество допустимых решений формируется согласно выражению (2.3). Решения, не принадлежащие множеству Υ, исключаются из дальнейшего вычислительного процесса

$$\Upsilon = \{x | \phi_r(x) = 0, \forall r \in R^{\text{жест}}\} \quad (2.3)$$

Для мягких ограничений каждому правилу $r \in R^{\text{мягк}}$ сопоставляется штрафное значение $\Delta_r^{(m)}$ по критерию m. Тогда совокупная надбавка по критерию m вычисляется согласно формуле (2.4)

$$P^{(m)}(x) = \sum_{r \in R^{\text{мягк}}} \phi_r(x) \cdot \Delta_r^{(m)}(x). \quad (2.4)$$

Тогда обобщенная целевая функция по критерию m имеет вид согласно выражению (2.5)

$$F^{(m)}(x) = c^{(m)T} x + P^{(m)}(x). \quad (2.5)$$

Для приведения разнонаправленных и разномасштабных критериев к единой шкале, для каждого критерия предварительно определяются минимальное и максимальное значения $F_m(x)$ (S_m^{min} и S_m^{max} соответственно) и вычисляется нормализованное значение согласно формуле (2.6):

$$F_{\text{норм}}^{(m)}(X) = \begin{cases} \dfrac{F^{(m)}(x) - S_m^{min}}{S_m^{max} - S_m^{min}}, \text{если } g_m = max, \\ \dfrac{S_m^{max} - F^{(m)}(x)}{S_m^{max} - S_m^{min}}, \text{если } g_m = min, \end{cases} ; F_m^{\text{норм}}(x) \in [0; 1], x \in \Upsilon. \quad (2.6)$$

Для получения итоговой величины качества решения, значения $F_m^{\text{норм}}(x)$, полученные для каждого критерия m, суммируются, как показано в формуле (2.7)

$$\omega(x) = \frac{\sum_{m \in M} F_{\text{норм}}^{(m)}(x)}{m}; \omega(x) \in [0; 1]. \quad (2.7)$$

В дальнейшем данная величина будет упоминаться под сокращением МКС (межкритериальная сумма). Наиболее качественным считается решение с наибольшей величиной МКС.

Наиболее приспособленным решением считается $x^* = \max\limits_x \omega(x)$.

2.2 Конфигурация генетического алгоритма для решения задач целочисленной условной оптимизации

Генетический алгоритм является одним из методов эволюционных алгоритмов, который был вдохновлен принципами естественного отбора и генетики. Решения, предлагаемые генетическим алгоритмом, эволюционируют через поколения с использованием операторов отбора, скрещивания и мутации, пока не будет достигнут один из критериев остановки. Генетический алгоритм имеет достаточно простые принципы, имеющие широкий простор для адаптации под поставленные задачи [30]:

– инициализация: создается начальная популяция случайных решений. Решения представлены в виде множества любой размерности и любых числовых форматов (бинарный, вещественный, целочисленные или их комбинация);

– оценка: каждая особь оценивается с помощью функции приспособленности, которая определяет качество решения. В контексте задачи ЦУО, функции приспособленности представляют собой сумму значений между точками в соответствии с матрицей смежности и вычисляемым критерием;

– отбор: выбираются особи для размножения на основе их значений приспособленности;

– скрещивание: комбинируются гены родительских особей для создания потомков;

– мутация: вносятся случайные изменения в гены потомков для поддержания разнообразия;

– замена: формируется новое поколение из текущих особей и их потомков;

– повторение: цикл продолжается до выполнения критерия остановки.

В целях обеспечения возможности решения задач ЦУО, генетический алгоритм был модифицирован таким образом, что значением

приспособленности является $\omega(x)$, вычисляемым в соответствии с формулой (2.6). Соответственно, на протяжении всего вычислительного процесса для каждого критерия хранятся и актуализируются значения s_m max и $s_m min$. Заранее стоит упомянуть, что особями подразумеваются решения (акторы), обозначаемые ранее переменной *x*.

Генерация особей происходит согласно формуле (2.8).

$$x = \{i_1, i_2, \dots, i_n\} \tag{2.8}$$

Где i – уникальная случайная точка, входящая в множество *V*.

Для того, чтобы удовлетворить фундаментальное ограничение существующей многокритериальной задачи ($\{l_1, l_2, \dots, l_n\} \in x$ уникальны) был выбран ограниченный перечень методов скрещивания, мутации и отбора.

Из методов скрещивания был выбран метод упорядоченного скрещивания (OX1) [21]. Пусть скрещиванию подлежат особи $x1$ и $x2$. Тогда принцип работы данного метода пошагово изложен в таблице 2.1.

Таблица 2.1 – Принцип работы метода OX1

№ шага	$x1$ / $x2$						Описание
	1	2	3	4	5	6	Индекс точки (i\j)
1	1	2	3	4	5	6	Начальное состояние
	4	2	6	3	1	5	
2			6	3	1		Применение двухточечного скрещивания по точкам *i = 3* и *j = 5*
			3	4	5		
3			6	3	1	2	Перенос значений слева-направо, начиная с точки *i - 1* к точке *j + 1*
			3	4	5	2	
4	4		6	3	1	2	После достижения края подмножества, начинает заполняться начало, уже имеющиеся значения пропускаются
	6		3	4	5	2	
5	4	5	6	3	1	2	Результат скрещивания
	6	1	3	4	5	2	

Далее будет дано описание методам мутации. Из известных методов мутации, выбор был остановлен на методах «Смены индексов» (далее – СИ) и «Реверса подмножества» (далее – РП).

Метод «Смены индексов» заменяет позиции двух случайных элементов множества друг с другом, как показано в формуле (2.9).

$$x_{\text{мут}} = \{x_1, x_2, \dots, x_{i-1}, x_j, x_{i+1}, \dots, x_{j-1}, x_i, x_{j+1}, \dots x_n\}, i > j, i \in \mathbb{Z}, j \in \mathbb{Z} \quad (2.9)$$

Где i, j – случайные числа.

Метод «Реверса подмножества» работает схожим образом: выбираются точки i и j, первая из которых является индексом начала, вторая длиной подмножества. Затем, выбранное подмножество заменяется в множестве на свой «реверсированный» вариант в соответствии с формулой (2.10).

$$x_{\text{мут}} = \{x_1, x_2, x_{i-1}, x_{i+j-1}, x_{i+j-2}, \dots, x_i, x_{i+1} \dots, x_n\}, \quad i - j > 0, i + j < n, i \in \mathbb{Z}, j \in \mathbb{Z} \quad (2.10)$$

Далее, будет дано описание используемых методов отбора. Каждый из методов отбора в конечном итоге полностью обновляет популяцию ρ в соответствии с правилами, изложенными в формуле (2.11).

$$\rho = \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n\}, \bar{x} \notin \emptyset, \quad (2.11)$$

где $\bar{x}$ – особь, прошедшая отбор и пересчет приспособленности (обновленная).

Принцип метода отбора «Рулетка» заключается в установлении вероятностей выбора той или иную особь в соответствии с его приспособленностью. Для вычисления вероятностей, следует вычислить суммарную приспособленность в соответствии с формулой (2.12).

$$\mathbf{x}_{\text{сум}} = \sum_{\mathbf{i=1}}^{\mathbf{v}} \boldsymbol{\omega}(\mathbf{x_i}) \quad (2.12)$$

где v – численность популяции; x_i – отдельно взятая особь (актор) по индексу i; $\omega(x_i)$ – функция приспособленности, вычисляемая по формуле (2.7).

Далее, для каждого индивидуума (актора) по формуле (2.13) вычисляется вероятность.

$$p_i = \frac{\omega(x_i)}{x_{\text{сум}}} \tag{2.13}$$

В контексте метода «рулетка», особь $\bar{x}$ будет формироваться в соответствии с уравнением (2.14).

$$\bar{x}_i = \begin{cases} x_i, & \text{если } j \le p_i \\ \emptyset, \text{если } j > p_i \text{ или } F(x_i) = \emptyset \end{cases}, \quad \text{где } i \in [1; v] \tag{2.14}$$

где j – случайное вещественное число в диапазоне [0;1]; t – численность популяции.

Метод стохастического отбора является методом, который выбирает определенное количество особей случайным образом, как это показано в уравнении (2.15).

$$\bar{x}_i = \begin{cases} x_i, & \text{если } j \ge 0{,}5 \\ \emptyset, & \text{если } j < 0{,}5 \end{cases}, \quad \text{где } i \in [1; v]. \tag{2.15}$$

На таблице 2.2 приведен общий список полученных модификаций ГА, используемых в дальнейшей работе.

Таблица 2.2 – Общий список полученных комбинаций параметров ГА

Метод скрещивания	Метод мутации	Метод отбора
OX1	РП	Рулетка
	СИ	Рулетка
	РП	Стохастический
	СИ	Стохастический
	РП	Турнирный
	СИ	Турнирный
	РП	Лучшие N
	СИ	Лучшие N

2.3 Модификация алгоритма муравьиной колонии для решения задач целочисленной условной оптимизации

Алгоритм муравьиной колонии является эволюционным алгоритмом, вдохновленным поведением муравьев при поиске пищи. Муравьи оставляют феромоновые следы, которые помогают колонии находить кратчайшие пути от источника пищи к муравейнику. Этот алгоритм показывает эффективность в решении задач комбинаторной оптимизации [31, 32, 33, 34] и, в частности, подходит для решения поставленной задачи ЦУО [35, 36, 37, 38].

На основе данного алгоритма, в рамках проводимого исследования был реализован эволюционный алгоритм, работающий согласно аналогичным принципам, но стремящийся максимально удовлетворить сразу несколько критериев (далее - МА). Реализовывать множество модификаций, на основе МА оказалось избыточным, поэтому было решено остановиться на одной реализации, поведение которой будет управляться при помощи параметров (коэффициентов эвристики).

Процесс построения решений основан на пошаговом выборе значений переменных. Для каждой пары (j, v) – индекса переменной и возможного значения – хранится феромон T_{jv} и эвристическая оценка η_{jv}. Эвристика определяется согласно формуле (2.16)

$$\eta_{jv} = \frac{1}{1 + \Delta F_{jv} + \psi_{jv}}, \tag{2.16}$$

где ΔF_{jv} – ожидаемое изменение целевой функции при присвоении $x_j = v$;

ψ_{jv} – оценка возможного нарушения ограничений.

Выбор значения осуществляется стохастически согласно формуле (2.17)

$$p_{jv} = \frac{[\tau_{jv}]^{\alpha}[\eta_{jv}]^{\beta}}{\sum_{(k,u)\in N}[\tau_{ku}]^{\alpha}[\eta_{ku}]^{\beta}}, \tag{2.17}$$

где N – множество допустимых вариантов; α, β – параметры важности феромона и эвристики.

После формирования допустимого вектора x вычисляется значение $F(x)$. Лучшие решения усиливают феромон согласно формуле (2.18)

$$\tau_{jv} \leftarrow (1-\rho)\tau_{jv} + \rho Q(x), \qquad Q(x) = \frac{1}{\omega(x)+\epsilon}, \tag{2.18}$$

где ρ – коэффициент испарения.

Таким образом, МА итеративно формирует допустимые целочисленные решения, комбинируя вероятностный поиск с учетом структуры целевой функции и ограничений ЦЛП, что обеспечивает согласованность алгоритма с математической моделью задачи.

Описание параметризации МА приведены в таблице 2.3. практическое влияние параметров друг на друга были проанализированы и описаны в источнике [23, 36]. Стоит отметить, что все приведенные ниже коэффициенты являются вещественными числами.

Таблица 2.3 – описание параметризации МА

	Краткое описание	Диапазон допустимых значений	Влияние на работу МА
α	Коэффициент чувствительности к феромону	$[0;\infty]$	При увеличении ускоряется сходимость, но увеличивается риск «застревания» на одном из субоптимальных решений
β	Коэффициент «жадности»	$[0;\infty]$	Увеличение позволяет быстро находить «перспективные» направления, замедляет сходимость
q	Коэффициент оставления феромонов	$(0;1]$	Увеличение способствует средоточию Н на качественных решениях, позволяет находить «компромиссные» решения, но имеет риск «застревания» на одном из субоптимальных решений
τ	Коэффициент испарения	$(0;1)$	Увеличение способствует поддержке разнообразия решений, снижению риска «застревания», но замедляет сходимость

2.4 Адаптация алгоритма пчелиной колонии под задачи целочисленной условной оптимизации

Алгоритм пчелиной колонии является эволюционным алгоритмом оптимизации, основанный на коллективном интеллекте роя пчел. Данный алгоритм имитирует процесс поиска пищи пчелами в природе. Они, взаимодействуя друг с другом, эффективно находят источники нектара, что позволяет использовать эту модель для решения сложных задач оптимизации. Основным преимуществом данного алгоритма заключается в его простоте и способности избегать локальных оптимумов благодаря сочетанию случайного поиска и целенаправленного улучшения решений [39, 40, 41, 42].

В рамках данной диссертационной работы, алгоритм пчелиной колонии был модифицирован и адаптирован под решение поставленных задач ЦУО. Далее, полученная модификация алгоритма пчелиной колонии будет упоминаться под сокращением «ПА». Как будет описано далее, некоторые «механики» ПА «унаследовал» от ГА.

Как известно, каждый итерационный шаг алгоритма пчелиной колонии состоит из двух основных фаз:

– исследование пчелами-работницами известных источников пищи;
– поиск пчелами-разведчиками новых источников пищи.

Аналогично ГА, свою работу ПА начинает с случайной генерации решений, результаты которой будут составлять популяции акторов, выполняющих роли пчел-работниц и пчел-разведчиков. Генерация решений производится в соответствии с формулой (2.6).

Механизм поиска пищи разведчиками работает аналогично механизму мутации у ГА. Соответственно, для этого использовались методы СИ и РП, описанные в формуле (2.9) и формуле (2.10) соответственно.

В ПА внедрен механизм выбора работниками решения, найденного разведчиками, с использованием вероятностной функции p, приведенной в формуле (2.19)

$$\rho(x) = \frac{\omega(x)}{\sum_{x \in B} \omega(x)}, \quad (2.19)$$

где B – множество разведчиков;$\omega(x)$ – функция вычисления МКС по формуле (2.7).

ПА можно параметризировать, задавая метод поиска пищи и долю работников. Увеличение доли позволяет исследовать узкий круг эффективных решений, способствуя его дальнейшему улучшению. Следовательно, это снизит разнообразие решений – меньше ресурсов будет выделено на поиск и разведку, что может привести к застреванию в локальных оптимумах, особенно, если начальная популяции была «неудачно» распределена. Уменьшение доли работников широко распределит популяцию по пространству решений, но при этом замедлит сходимость.

2.5 Модификация алгоритма имитации отжига для решения задач целочисленной условной оптимизации

Для решения поставленной задачи ЦУО метод имитации отжига [43, 44] был модифицирован. В дальнейшем реализованная модификация, основанная на данном методе, будет упоминаться под фразой «ИО». Стоит отметить, что для получения нового решения, ИО «наследует» методы, аналогичные методу мутации в ГА и методу поиска у ПА: СИ и РП.

Для принятия решения о переходе от текущего решения к новому, необходимо вычислить изменение агрегированной стоимости согласно формуле (2.20).

$$\Delta = \omega(\mathrm{x`}) - \omega(\mathrm{x}), \quad (2.20)$$

где x – текущее решение; $x`$ – предлагаемое решение (полученное в результате мутации).

Вероятностный коэффициент P вычисляется согласно формуле (2.21).

$$P = e^{\left(\frac{\Delta}{T}\right)}, \tag{2.21}$$

где T – значение температуры на момент работы алгоритма.

В уравнении (2.22) показан принцип замены текущего решения на новое, в том числе при отрицательном или нулевом значении Δ. Т.к. мерой эффективности решения является максимизация МКС, положительное значение Δ будет свидетельствовать об улучшении качества решения и замене текущего решения на полученное новое.

$$x = \begin{cases} x`, & \text{если } \Delta > 0 \\ x`, & \text{если } \Delta \leq 0 \text{ и } q \leq P \\ x, & \text{если } \Delta \leq 0 \text{ и } q > P \end{cases}, \quad \text{где } q, P \in [0; 1];\ q, P \in \mathbb{R} \tag{2.22}$$

где q – сгенерированное случайное число, P – пороговое значение (параметр);

ИО принимает на вход параметры начальной температуры (T_0), порогового значения температуры (T_1) и коэффициента охлаждения (η). Для того, чтобы интерфейс алгоритма имел максимальное сходство с МА, ГА и ПА, была внедрена возможность задавать количество итерационных шагов. Таким образом, работа ИО будет прекращаться либо при достижении условия $T \leq T_1$ либо после прохождения заданного количества шагов. На таблице 2.4 дано описание параметризации ИО.

Таблица 2.4 – Описание параметризации ИО

	Краткое описание	Примеры значений	Влияние на работу ИО
T_0	Начальная температура	100; 500; 1000	Высокое значение допускает большее число случайных переходов, что замедляет сходимость, но помогает избежать локальных оптимумов
T_1	Конечная температура	0,1; 0,001; 0,0001	Наиболее интенсивное исследование пространства решений, но увеличение вычислительной сложности
η	Коэффициент охлаждения	0,95; 0,99; 0,8	Приближение значения к 1 замедляет время работы и делает поиск более интенсивным; уменьшение значение может привести к ускорению работы, но застреванию в локальных оптимумах

2.6 Проектирование и спецификация средств формализации представления ограничений

Как было описано ранее, для поставленной задачи ЦУО могут быть заданы ограничения, которые являются логическими выражениями, при истинности которых санкции будут применяться к промежуточным «покритериальным» расчетам вплоть до исключения решения. Такие ограничения записываются в виде предикатов синтаксисом, который обладает краткостью, не потребует длительного изучения и имеет возможность описывать логику любой уровни сложности.

В рамках данной работы был разработан синтаксис правил формализации представления ограничений. Он был основан на польской записи (префиксной нотации) [45, 46, 47], которая обеспечивает эффективную оценку выражений без необходимости учета приоритета. Иной причиной выбора польской нотации является ее краткость написания. На таблице 2.5 показаны операторы формализации представления ограничений.

Таблица 2.5 Операторы формализации представления ограничений

Обозначение	Краткое описание	Шаблон	Примеры	
@	От точки до точки по критерию	@ [Точка1] [Точка2] [Оператор сравнения или ~] [Значение или диапазон]	@ A B время > 300	@ B C время ~ [275,375]
#	Позиция точки по номеру	# [Точка] [Оператор сравнения или ~] [Значение или диапазон]	# A = 1	# A ~ [5,7]
>>	Следование точки за точкой	>> [Точка1] [Точка2]	>> D I	>> M A
>>>	Следование точки непосредственно за точкой	>>> [Точка1] [Точка2]	>>> A N	>>> NA
&	Логическое «И»	[Выражение1] & [Выражение2]	@ A B время > 300 & # A =1	
|	Логическое «ИЛИ»	[Выражение1] | [Выражение2]	@ A B время > 300 | # A = 1	
!	Логическое отрицание	!([Выражение])	!(# A > 1)	A B дистанция != 125

Стоит упомянуть, что помимо операторов сравнения у первых двух команд можно указывать оператор вхождения в диапазон, требующий наличия символа «~» перед ним, и состоящий из круглых или квадратных скобок и значений. Данный оператор способен обрабатывать диапазоны, подразумевающие строгое и нестрогое соответствие, а также, отсутствие одного из значений, что будет означать «от начала» и «до конца» соответственно. Варианты использования оператора диапазона приведены на таблице 2.6.

Таблица 2.6 Варианты использования оператора диапазона

Пример	Описание
[1, 2]	От 1 включительно до 2 включительно
(1, 2]	От 1 не включительно до 2 включительно
[1, 2)	От 1 включительно до 2 не включительно
(1, 2)	От 1 не включительно до 2 не включительно
[, 2]	От 0 включительно до 2 включительно
(, 2)	От 0 не включительно до 2 не включительно
(1,]	От 1 не включительно до конца включительно
(1,)	От 1 не включительно до конца не включительно

Логическое выражение разделяет фраза «->», справа от которой описаны санкции, которые будут применены к решению. Санкции могут описаны в виде пар «имя критерия - значение», перечисленных через запятую (например «time:300, distance:-150») или в виде ключевого слова «exc», что будет означать полное исключение решения из дальнейших расчетов. В случае написания санкции «парами», значения будут прибавлены к промежуточным результатам решения, а сумма вычисленного результата и санкций будет учтена при оценке качества решения на общих основаниях [48].

Пример ограничения в контексте распределения взаимосвязанных вычислительных процессов на доступных узлах кластера приведен ниже:

@ P1 P2 задержка > 100 & >> N3 N5 & # N3 = [2,4] -> время:1500;отказоуст:-10

Интерпретация: если между процессами P1 и P2 средняя задержка связи превышает 100, при этом узел N3 предшествует N5 в порядке маршрутизации, и N3 занимает позицию в диапазоне [2;4], то конфигурация считается неэффективной и получает штраф 1500 к времени выполнения и -10 к отказоустойчивости.

@ P1 P2 задержка соотносится к сетевым характеристикам, конструкция >> N3 N5 описывает топологическую зависимость между узлами, конструкция # N3 = [2,4] позволяет учесть позицию узла в вычислительном графе, например, когда средние узлы перегружаются чаще краевых.

2.7 Сравнительная характеристика выбранных эволюционных алгоритмов

Сравнительная характеристика эволюционных алгоритмов, составленная на основе источников [26, 27, 28], на основе которой был сделан выбор в пользу тех или иных алгоритмов для исследования, показана на таблице 2.7.

Каждый из описанных алгоритмов имеет свои особенности, полученные от аналогичных «физических» процессов, что позволяет охватить как глобальный, так и локальный поиск оптимальных решений. Это обеспечивает более широкий взгляд на решение задачи т.к. разные алгоритмы по-разному балансируют между «интенсификацией» и «экстенсификацией» исследования пространства решений.

Таблица 2.7 – сравнительная характеристика алгоритмов

Алгоритм	Преимущества	Недостатки	Модификации и параметры
Муравьиный	Использует коллективную память для усиления перспективных решений Хороший баланс между интенсификацией и экстенсификацией	Возможно застревание в локальных оптимумах Чувствителен к настройке параметров	Коэффициент чувствительности к коллективной памяти, коэффициент «жадности», интенсивность запоминания, интенсивность забвения

Продолжение таблицы 2.7

Пчелиный	Высокая диверсификация поиска Сочетание глобального и локального поисков	Может потребовать большего числа итераций	Доля рабочих, метод мутации (поиска)
Генетический	Широкий охват пространства решений Гибкость в представлении и обработке акторов	Относительно медленная сходимость Риск застревания в локальных оптимумах	Метод мутации, шанс мутации, метод отбора
Имитация отжига	Простота реализации Способность выхода из локального оптимума Адаптация к сложным пространствам в целом	Медленная сходимость Требует «тонкой» настройки графика охлаждения	Начальная температура, конечная температура, коэффициент охлаждения, метод мутации (поиска)
Другие (например, роя частиц)	Быстрая сходимость Простота реализации Небольшое число параметров	Сильная зависимость от начальных условий Риск застревания в субоптимальных решениях	Скорость поиска, коэффициенты инерции и обучения

В дополнение стоит отметить, что выбор алгоритмов в пользу МА, ГА, ПА и ИО был сделан из расчета, что эти алгоритмы будут проявлять свои преимущества и компенсировать недостатки друг друга при решении задач ЦУО с интегрированным механизмом переключения алгоритма. В контексте сравнения алгоритмов друг с другом можно выделить следующие моменты:

– все эволюционные алгоритмы требуют настройки параметров, однако ГА и МА намного чувствительнее к настройке, чем ИО и ПА;

– благодаря своему «одиночному» подходу, ИО часто является более быстрым, чем другие эволюционные алгоритмы, однако это имеет меньшую устойчивость к застреванию в субоптимальных решениях;

– ПА может иметь скорость сходимости, сравнимую с ГА, но в качестве решения, ПА может опережать ГА благодаря механизму работы разведчиков;

– МА имеет достаточно высокий разброс скорости сходимости: часто она медленнее, чем у ИО, но может быть быстрее, чем у ГА в больших данных, благодаря наличию механизма коллективной памяти;

– ГА может иметь низкую скорость, особенно при больших данных, что связано с необходимостью многократных операций скрещивания и мутации.

2.8 Проектирование модуля интеллектуального выбора стратегии для решения задач целочисленной условной оптимизации

В рамках данной работы под стратегией решения задачи ЦУО понимается упорядоченное множество определенных конфигураций ЭА [49, 50, 51]. Каждая конфигурация ЭА определяется как объект $conf_j$, которому присвоен уникальный идентификатор. Множество всех допустимых конфигураций образует пространство конфигураций в соответствии с выражением (2.23).

$$\mathcal{C} = \{conf^{(1)}, conf^{(2)}, \dots, conf^{(N_C)}\}, \tag{2.23}$$

где $conf^{(ID)}$ – конфигурация с идентификатором ID; N_C – общее множество всех предопределенных конфигураций.

Стратегия решения задачи представляется в виде упорядоченного множества идентификаторов конфигураций ЭА, как показано в выражении (2.24). Порядок элементов определяет приоритет применения соответствующих конфигураций в процессе решения задач ЦУО

$$S = (ID_1, ID_2, \dots, ID_M);\ ID_j \in \{1,2, \dots, N_C\}, \tag{2.24}$$

где M – количество конфигураций в стратегии, ID_j – идентификатор конфигурации на позиции j.

Критерий переключения между конфигурациями в стратегии определяется отсутствием улучшения целевой функции на протяжении $n_{\text{стаг}}$ итераций в соответствии с формулой (2.25)

$$\Delta = \left| f\left(x_{\text{лучш}}^{(t)}\right) - f\left(x_{\text{лучш}}^{(t-n_{\text{стаг}})}\right) \right| < \epsilon, \tag{2.25}$$

где $x_{\text{лучш}}^{(t)}$ – лучшее решение на итерации t; ϵ – порог стагнации.

Интеллектуальный модуль реализует отображение [51] в соответствии с выражением (2.26).

$$f: \mathcal{R} \to \{1,2, \dots, N_C\}^M, \tag{2.26}$$

где $\mathcal{R}$ – множество формализованных представлений ограничений, f – функция, аппроксимируемая моделью трансформерного типа, формирующая массив конфигураций эволюционных алгоритмов на основе структурного анализа ограничений задачи.

Обучение интеллектуального модуля осуществляется на наборе задач ЦУО, для которых эмпирически получены эффективные множества конфигураций ЭА [53].

Каждому обучающему примеру соответствует пара $(Z^{(i)}, S^{(i)})$, где $Z^{(i)}$ – контекстное представление задачи, сформированное на основе системы ограничений, а $S^{(i)} = (c_1^{(i)}, \dots, c_M^{(i)})$ – целевое множество конфигураций. Предсказание стратегии осуществляется согласно выражению (2.27).

$$\hat{S}^{(i)} = g_\theta\left(Z^{(i)}\right), \tag{2.27}$$

где g_θ – интеллектуальная трансформерная модель.

Для повышения эффективности и устойчивости вычислительного процесса предложен интеллектуальный модуль, обеспечивающий автоматическое определение типа оптимизационной задачи и выбор конфигурации эволюционного алгоритма [54]. Для ее работы, каждое ограничение *r* представляется в виде последовательности токенов $r = \{t_1, t_2, \dots, t_L\}$, где каждый токен принадлежит словарю $t \in v$, включающему операторные элементы, точки графа и числовые значения интервалов. Для каждого токена задается обучаемое векторное представление $e_i = E(t_i), e_i \in \mathbb{R}^d$. Для учета относительного порядка используется позиционное встраивание $\tilde{e}_i = e_i + p_i$, где p_i – фиксированный синусоидальный вектор длиной *d*.

Контекстное представление формируется с использованием «самовнимания» [55]. Для каждого токена вычисляются запрос, ключ и значение согласно выражению (2.28).

$$\mathrm{q_i} = W_Q \tilde{e}_i; \quad k_i = W_K \tilde{e}_i; \quad \upsilon_i = W_V \tilde{e}_i \tag{2.28}$$

Матрица весов внимания и контекст вычисляются согласно выражению (2.29).

$$\alpha_{ij} = \frac{e^{\left(\frac{q_i k_j^T}{\sqrt{d}}\right)}}{\sum_{m=1}^{L} e^{\left(\frac{q_i k_m^T}{\sqrt{d}}\right)}}, h_i = \sum_{j=1}^{L} \alpha_{ij} \upsilon_j. \tag{2.29}$$

Для повышения качества работы интеллектуальной модели используется несколько независимых механизмов «внимания». Результирующее представление токена формируется путем объединения всех компонент и последующей линейной трансформации согласно выражению (2.30)

$$\tilde{h}_i = W_O[h_1^{(1)}, h_2^{(2)}, \dots, h_i^{(H)}], \tag{2.30}$$

где W_O – матрица преобразования, $[h_1^{(1)}, h_2^{(2)}, \dots, h_i^{(H)}]$ – склейка отдельных компонент в единый вектор, H – количество «голов» внимания.

Итоговое представление всего ограничения (z_r) получается либо усреднением, либо суммированием, либо обучаемым взвешиванием. Итоговое представление задачи определяется в соответствии с выражением (2.31)

$$Z = \Psi\left(z_{r_1}, \dots z_{r_K}\right), Z \in \mathbb{R} \tag{2.31}$$

где Ψ – функция объединения всех ограничений, $Z \in R^{d_Z}$ – векторное представление задачи, K – количество ограничений в задаче, z_{r_k} – представление k-го ограничения, d_Z – размерность представления задачи.

2.9 Выводы

В рамках этого раздела были получены следующие результаты:

1. Разработана математическая модель задачи целочисленной условной оптимизации, включающая формализацию множества перестановок, многокритериальную систему оценки с нормализацией разнонаправленных критериев и механизм интеграции штрафных функций для жестких и мягких ограничений, что обеспечивает единую метрику качества решений через межкритериальную сумму.

2. Проведен системный анализ применимости эволюционных алгоритмов к задачам целочисленной условной оптимизации, выявлены их преимущества и недостатки, определены направления модификации, что обеспечило теоретическое обоснование выбора алгоритмов и стратегий их применения.

3. Модифицированы и адаптированы эволюционные алгоритмы (генетический, муравьиный, пчелиный и имитации отжига) для решения задач целочисленной условной оптимизации. В алгоритмы интегрирован механизм верификации промежуточных решений и реализован интерфейс взаимодействия с системой выбора и переключения алгоритмов.

4. Проведен сравнительный анализ характеристик эволюционных алгоритмов, выявивший их комплементарные свойства: различную чувствительность к настройке параметров, разную скорость сходимости и устойчивость к «застреванию» в локальных, субоптимальных решениях, что обосновывает целесообразность их вариативного применения в системе.

5. Спроектирован интеллектуальный модуль выбора стратегии решения задач целочисленной условной оптимизации на основе трансформерной архитектуры, реализующий отображение формализованных ограничений в упорядоченное множество конфигураций эволюционных алгоритмов с критериями переключения на основе стагнации улучшений.

6. Разработана система токенизации и контекстного представления ограничений с использованием векторного встраивания, позиционного

кодирования и многоголового внимания, что позволяет автоматически извлекать структурные зависимости между ограничениями и формировать эффективную стратегию поиска.

ГЛАВА 3. РАЗРАБОТКА АЛГОРИТМИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РЕШЕНИЯ ЗАДАЧ ЦЕЛОЧИСЛЕННОЙ УСЛОВНОЙ ОПТИМИЗАЦИИ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ

3.1 Описание структуры системы

Структура алгоритмического обеспечения с верификацией промежуточных решений показана на рисунке 3.1.

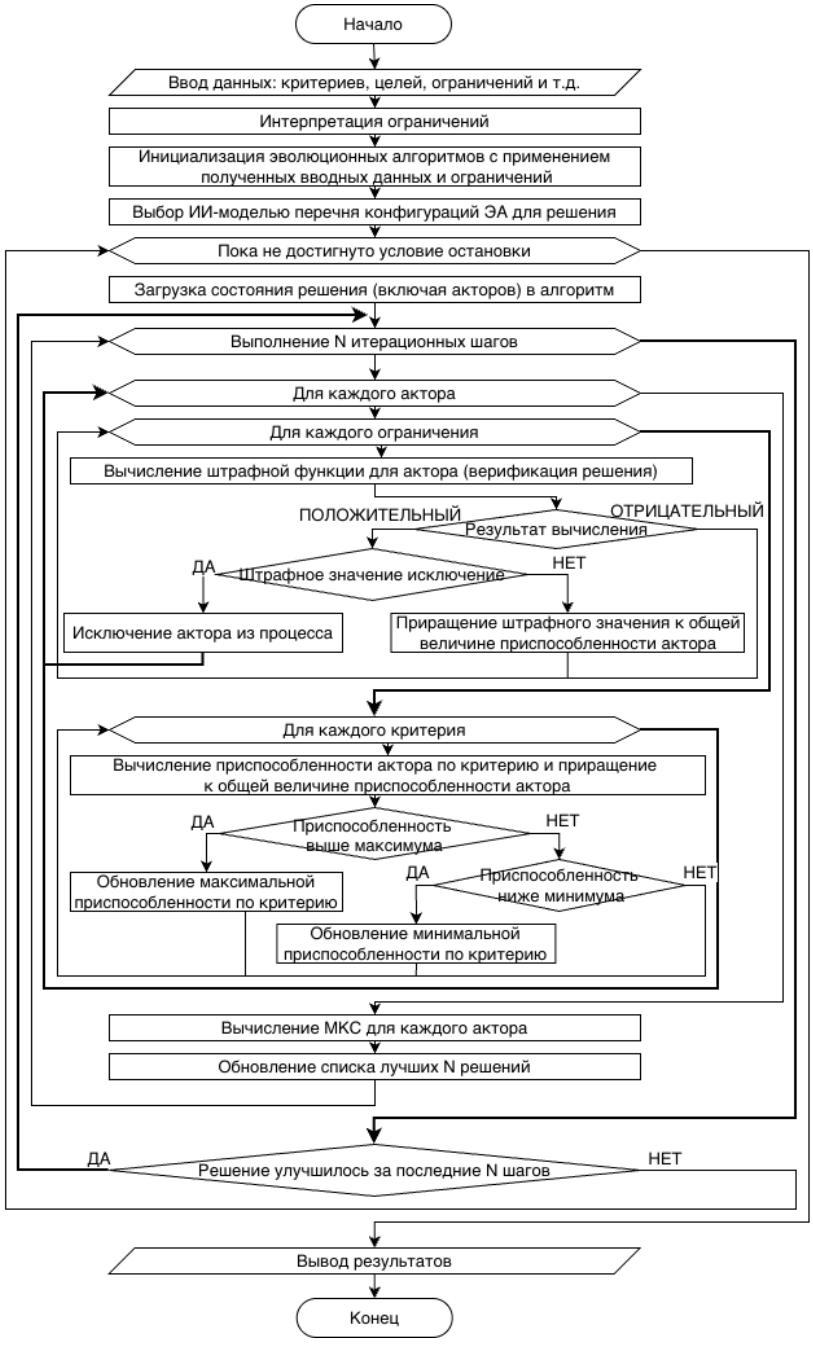


Рисунок 3.1 – Структура алгоритмического обеспечения с верификацией промежуточных решений

Последовательность этапов формирует сквозной цикл обработки входных данных, контроля ограничений, оценки приспособленности и динамического управления конфигурациями вычислительного процесса. Алгоритм инициируется вводом исходной информации, включающей значение целевых параметров, критериев оптимальности и полный перечень ограничений. На основе этих данных выполняется интерпретация ограничений, в ходе которой описание правил преобразуется в формальную структуру, предназначенную для дальнейшей автоматизированной проверки допустимости решений. Затем осуществляется инициализация эволюционных алгоритмов с учетом интерпретированных ограничений и параметров задачи. На этом же этапе интеллектуальный модуль избирает эволюционную стратегию, обеспечивающую соответствие начальных настроек спецификации задачи.

После настройки системы начинается итерационная стадия поиска решения, продолжающаяся до выполнения условия остановки. На каждом цикле восстанавливается текущее состояние решения, включая параметры акторов как абстракций испытуемых решений. После выполнения итерационного шага осуществляется процедура верификации промежуточных результатов.

Проверка корректности акторов проводится последовательно по каждому ограничению. Каждый актор «прогоняется» через штрафные функции, в результате чего для каждого из них вычисляется штрафное значение или исключение из дальнейшего процесса. Штрафные значения аккумулируются и встраиваются в общую величину приспособленности актора.

После проверки акторов на соответствие ограничениям выполняется этап оценки приспособленности по целевым критериям. Для каждого актора рассчитываются значения приспособленности по отдельным критериям, которые затем интегрируются в совокупную меру качества решения. Одновременно производится актуализация минимальных и максимальных значения приспособленности, необходимых для нормализации и последующего вычисления МКС. Эти операции обеспечивают корректное сравнение акторов

между собой и формируют механизм контроля границ значений критериев в процесс эволюции.

По завершении вычисления приспособленности определяется МКС для каждого актора. Система выбирает лучшие решения, обновляя внутренний список оптимальных акторов. Затем анализируется динамика изменения качества решений за последние несколько итерационных циклов. Если обнаружено улучшение, алгоритм продолжает работу, при отсутствии положительной динамики выполняется его переключение или переход к заключительным операциям. После достижения условия остановки производится вывод оптимальных решений и формирование итогового результата.

Далее, был определен перечень модулей, подлежащих реализации, а также связь между ними [55]. Результаты данного процесса приведены в виде структурной схемы в рисунке 3.2. Для взаимодействия модулей между собой, для многих из них предусмотрены блоки адаптеров, которые выполняют как роль интерфейса, так и роль инкапсуляции. Ниже приведено описание алгоритмов:

Модуль обработки входных данных отвечает за работу с данными, которые пользователь или система подает на вход системе, за настройку вычислений и компиляцию ограничений. Он содержит подмодули обработки параметров вычислений и обработчик постановок задач. Адаптер данного модуля также управляет работой модули настройки вычислений. Он в свою очередь отвечает за применение настроек, которые могут быть получены на вход опционально: организация параллельной вычислительной работы алгоритмов, установления условий остановки, а также установление ограничений к вычислительным ресурсам и времени.

Другим модулем, управляемым адаптером работы с входными данными, является интерпретатор языка ограничений. Данный модуль анализирует синтаксис языка ограничений, лексику, семантику и преобразовывает текст в машинный язык, который в дальнейшем интегрируется в работу эволюционных алгоритмов и вспомогательных методов.

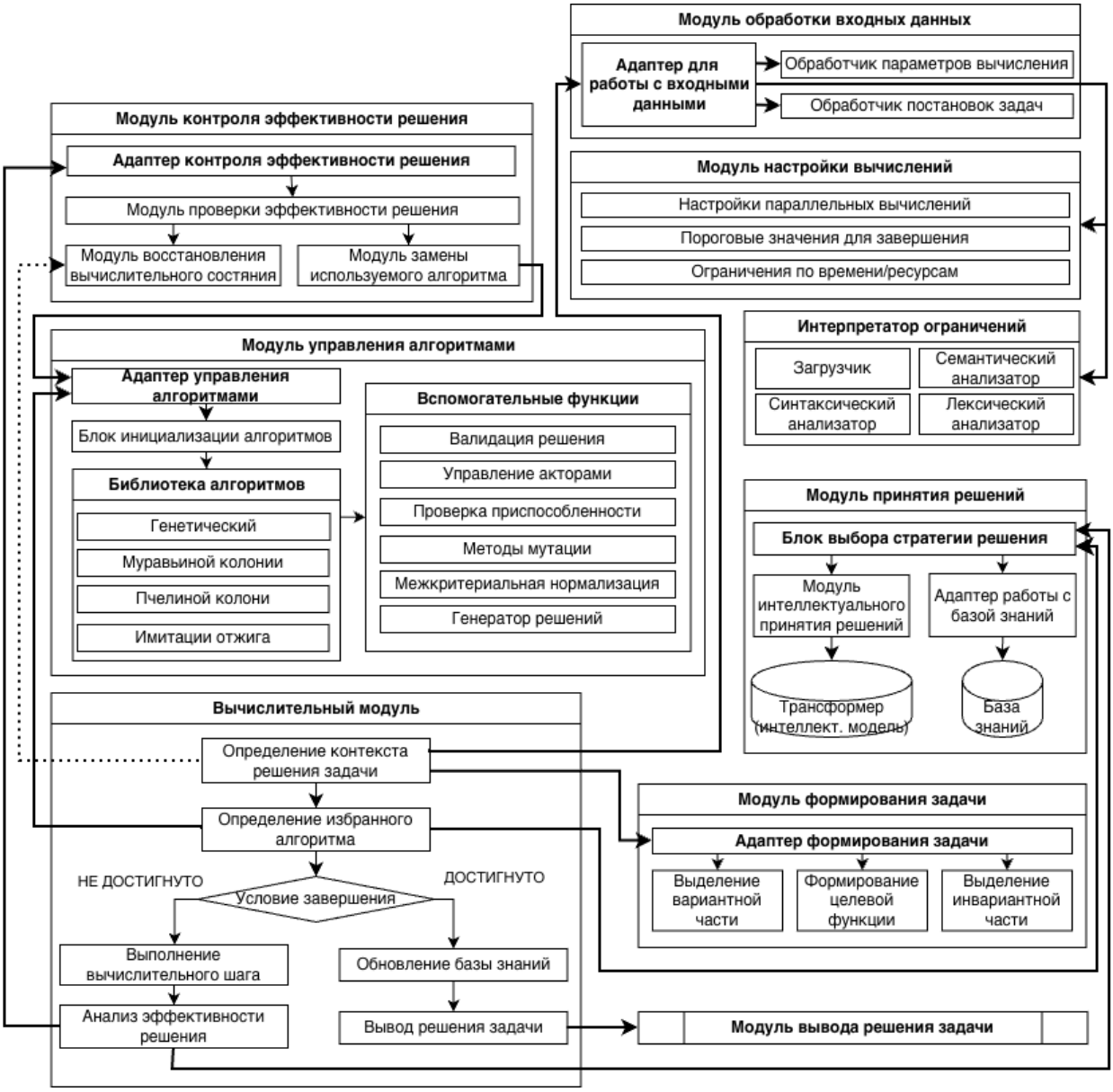


Рисунок 3.2 – Структура адаптивной интеллектуальной системы целочисленной условной оптимизации

Другим модулем является модуль принятия решений, основным блоком которого является блок выбора стратегии решения. Основными функциями данного модуля является долгосрочное хранение данных в виде БЗ и взаимодействие с заранее обученной ИИ-моделью. Кроме того, данный модуль определяет наличие решения ранее решенной аналогичной задачи в базе знаний.

Модуль формирования задачи выполняется после модуля обработки выходных данных. У полученных критериев, данный модуль выделяет

вариантную и инвариантную части, формирует целевую функцию и интегрирует опциональные параметры и ограничения. Данная целевая функция поставляется используется в алгоритмах для сравнения эффективности полученных решений.

Модуль управления алгоритмами является модулем, отвечающим непосредственно за работу и управление эволюционными алгоритмами. Он содержит блок инициализации алгоритмов, непосредственно реализации МА, ГА, ПА и ИО, а также вспомогательные функции, широко используемые во всех алгоритмах.

Одним из модулей, имеющих достаточно сильную связь с другими, является модуль контроля эффективности решения. Помимо проверки прогресса решения на необходимость в смене алгоритма, данный модуль занимается восстановлением вычислительного контекста, также проводит низкоуровневые мероприятия по смене алгоритма: извлечение контекста, уничтожение инстанса алгоритма, сборка мусора и т.д.

Основным модулем является вычислительный модуль. Он использует в своей работе почти все модули, существующие в системе. Содержимое модуля описано в виде алгоритмической схемы, содержащей основные части работы модуля: восстановление вычислительного контекста, определение избранного алгоритма, выполнение вычислительного шага и анализ эффективности или обновление базы знаний или передача результатов задачи модулю вывода. В свою очередь, модуль вывода решения задачи отвечает за структуризацию вывода – агрегацию, фильтрацию и общее приведение результатов вычисления в структурированный и читабельный вид.

Декомпозиция блока формирования задачи приведена на рисунке 3.3.

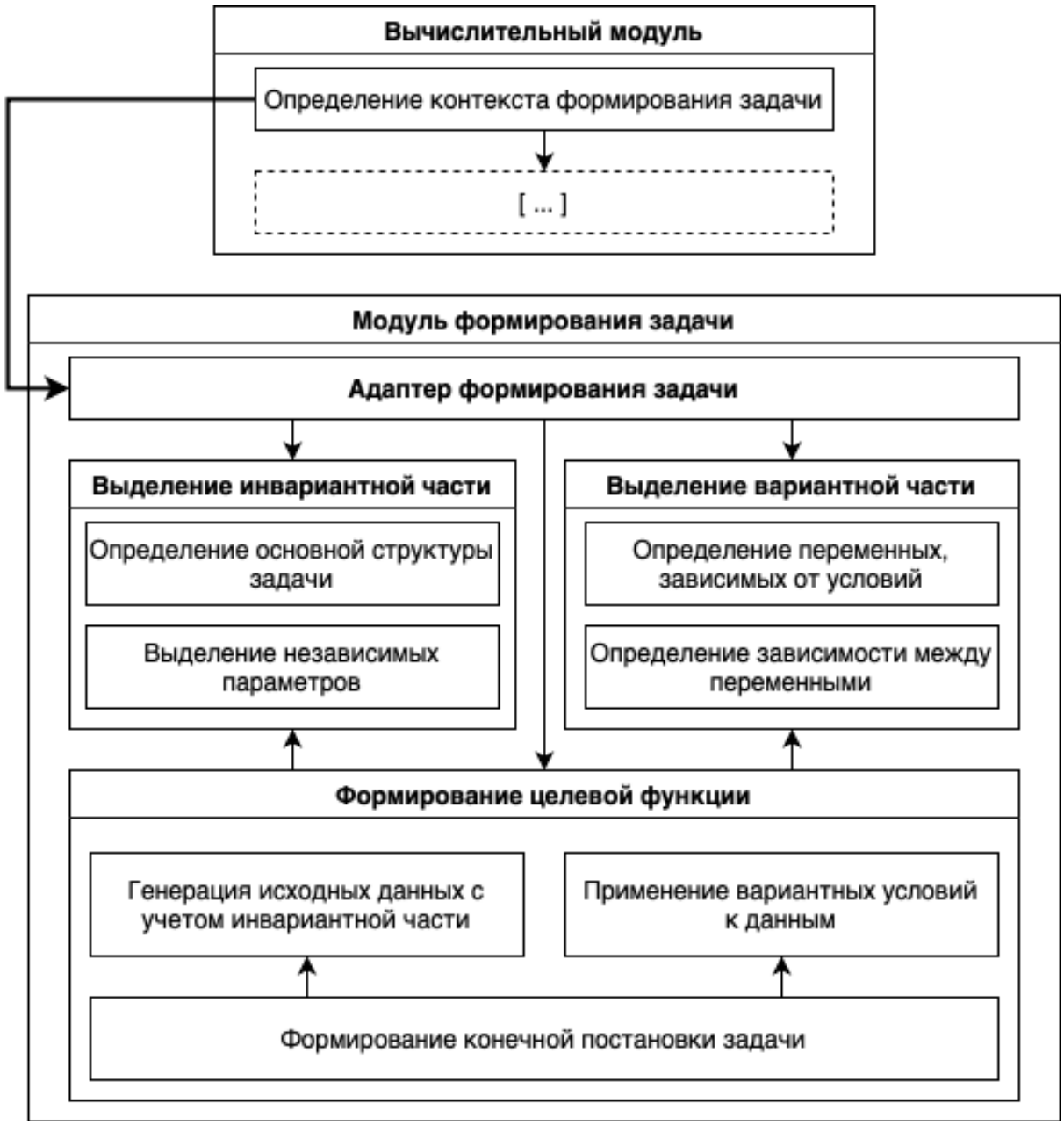


Рисунок 3.3 – Структурная схема (декомпозиция) модуля формирования задачи

Данный модуль имеет на входе следующие данные, описывающие задачу: ограничения, объекты критериев (наименование, цель, матрица смежности). В рамках инвариантной части, модуль выполняет следующую работу:

– определяет основную структуру задачи (например, модель графа, модель состояний, системы уравнений и пр.);

– выделяет параметры, не зависящие от условий задачи;

Другим важным блоком является выделение вариантной части, который в свою очередь занимается определением изменяемых переменных и их зависимостью друг от друга.

Основным блоком данного модуля является формирование целевой функции. Для ее получения требуется предварительно сгенерировать исходные данные с учетом инвариантной части, а затем применить к исходным данным и вариантную часть. На выходе блок выдает конечную постановку задачи в виде целевой функции (*f(x)*), которая будет выполнять оценочную роль для получаемых алгоритмами решений.

3.2 Комплексный анализ и выбор технологий для задач целочисленной условной оптимизации

Для выбора языка программирования и других технологий, проведен ряд мероприятий, представляющие собой сравнительный анализ языков программирования.

Исследуемая в данной работе целочисленная задача оптимизации является вычислительно сложной задачей, требующей высокой производительности и надежности, особенно, если система включает механизм переключения алгоритмов для адаптации к различным сценариям. Rust, как системный язык программирования, был выбран из-за своей способности обеспечивать высокую скорость выполнения, высокую ставку на безопасность (как с точки зрения типов, так и памяти), а также общую гибкость в разработке (в частности, выбора парадигмы). Ряд других преимуществ, оказавших влияние на конечный выбор, будут описаны далее, в процессе сравнительного анализа языков программирования.

Для поставленных задач язык программирования C++ был бы «традиционным» выбором в силу требований к производительности. Однако, согласно источникам [57, 58, 59] сравнение производительности показывает, что Rust составляет конкуренцию C++ в производительности за счет оптимизации компилятора. Кроме того, написание параллельного безопасного кода в Rust

намного проще, чем в C++, что открывает возможность нетрудоемкой интеграции параллелизма в работу эволюционных алгоритмов. На основе полученных результатов сравнительная характеристика Rust и C++ отображена на таблице 3.1.

Таблица 3.1 – Сравнение языков программирования Rust и C++

Характеристика	Rust	C++
Производительность	Высока, сравнима с C++	Высокая
Безопасность памяти	Высокая	Средняя, ручное управление
Параллелизм	Простой и безопасный	Требует большого объема знаний и осторожности
Гибкость кода	Высокая благодаря наличию трейтов и дженериков	Требует соблюдения принципов ООП

В настоящее время язык программирования Python имеет достаточно высокую популярность и множество библиотек, в том числе подходящих для научных вычислений или решения задач оптимизации [60, 61]. Основным недостатком, из-за которого выбор был сделан в пользу Rust, является низкая скорость работы, вследствие чего интенсивные задачи, имеющие прямое или косвенное отношение к оптимизации, работают достаточно медленно. Это обусловлено тем, что код Python интерпретируем. Rust, напротив, является компилируемым языком и проводит ряд оптимизаций кода в момент компиляции.

Однако, в рамках данной работы Python широко используется для проведения мероприятий, связанных с обучением ИИ-модели для переключения алгоритмов. В среде Python были произведены сбор, подготовка данных, их визуализация, обучение модели и анализ результатов. Для этого

использовались Python-библиотеки Pandas [62, 63], matplotlib, seaborn [64, 65, 66, 67], Scikit-learn [68, 69].

Языки программирования Java и C# также являются сильно типизированными и мультиплатформенными, но обладают сборщиком мусора, который может приводить к задержкам, что критично для скорости работы системы. Как описано в статье [70], Rust более эффективно управляет памятью без накладных расходов, что делает его наиболее предпочтительным для высокопроизводительного инструментария и оптимизации, в частности.

Также, в разработке системы использовались некоторые Rust-библиотеки. В разработке алгоритмов широко всего использовались библиотеки rand [71] и random_choice [72], которые предоставляют ряд методов по генерации случайных чисел, перемешивания множеств и случайного вероятностного выбора элементов множеств. Для оформления решений, их преобразования в формат JSON [73, 74], простой для чтения людьми и другими программами, использовалась библиотека serde с трейтом serde_json [75, 76], которая специализируется на сериализации и десериализации данных в различные структуры.

3.3 Описание правил базы знаний для взаимодействия с процессом целочисленной условной оптимизации

В интеллектуальной подсистеме, обеспечивающей решение задач целочисленной условной оптимизации, база знаний (БЗ) функционирует как специализированный модуль долговременного и кратковременного хранения данных. Её назначение заключается в структурировании ранее полученной вычислительной информации, обеспечении многократного использования результатов, сокращении временных и ресурсных затрат, а также формировании статистических признаков для обучаемой модели выбора конфигураций алгоритмов.

БЗ функционирует в двух режимах:

– долгосрочное хранилище, обеспечивающее сохранение формальных подписей задач и соответствующих решений;

– краткосрочный вычислительный кэш, предназначенный для хранения промежуточных результатов функций и оценок.

Обработка данных осуществляется посредством набора правил вывода, формализующих условия сохранения, обновления и извлечения информации:

ЕСЛИ
[процесс решения завершён]
И [решение прошло проверку корректности]
И [решение предназначено для вывода пользователю или дальнейшего анализа]
ТО
[сохранить запись вида (формальная подпись задачи, решение, использованная конфигурация алгоритмов, метаданные вычислений)].

ЕСЛИ
[поступила новая задача с формальной подписью S]
И [в базе знаний существует запись (S, решение)]
И [запись является актуальной и непомеченной как устаревшая]
ТО
[вернуть сохранённое решение].

ЕСЛИ
[инициировано формирование входных признаков для интеллектуальной модели]
И [в базе знаний хранится статистика эффективности конфигураций для подписи S]
ТО
[предоставить статистические данные модели для включения в признаки выбора конфигурации].

ЕСЛИ
[вызвана вычислительная функция F]
И [входные данные функции обозначены как X]
И [в базе знаний отсутствует запись (текущая задача, F, X, результат)]
ТО
[сохранить (текущая задача, F, X, вычисленный результат, отметка времени)].

ЕСЛИ
[вызвана вычислительная функция F]

И [входные данные функции обозначены как X]
И [в базе знаний существует запись (текущая задача, F, X, результат R)]
И [результат R не отмечен как просроченный или недействительный]
ТО
[вернуть значение R без повторного вычисления].

ЕСЛИ
[выполнен итерационный шаг алгоритма]
И [значения целевых функций или штрафов изменились]
ТО
[сохранить промежуточные данные в краткосрочный кэш как (итерация, данные, оценка качества решения)].

ЕСЛИ
[решение задачи завершено]
И [при решении применялась конфигурация C]
И [имеются данные о производительности или сходимости конфигурации C]
ТО
[обновить статистические показатели конфигурации C для данной подписи задачи].

ЕСЛИ
[вызвана функция F высокого уровня]
И [на вход функции передано решение R]
И [в базе знаний отсутствует запись (подпись задачи, F, R)]
И [функция подлежит кешированию согласно регламенту]
ТО
[сохранить (подпись задачи, F, R, результат вычисления G)].

ЕСЛИ
[вызвана функция F высокого уровня]
И [на вход передано решение R]
И [в базе знаний существует запись (подпись задачи, F, R, результат G)]
ТО
[вернуть результат G].

3.4 Реализация интерпретатора правил формализации представления ограничений

Схема алгоритма формализации представления ограничений на основе правил приведена на рисунке 3.4.

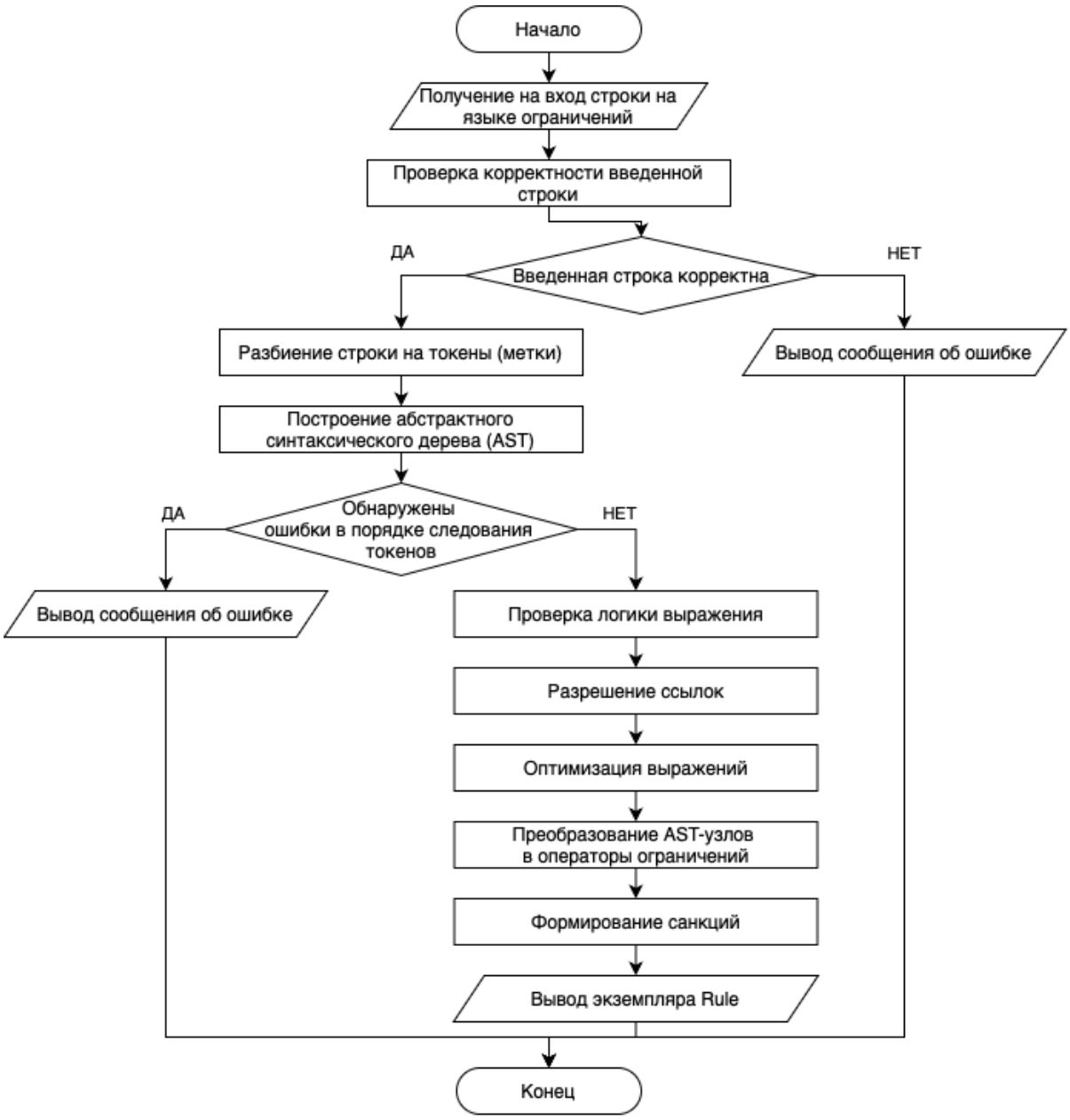


Рисунок 3.4 - Схема алгоритма формализации представления ограничений на основе правил

Ниже описаны основные этапы интерпретации [77]:

– лексический анализ: входная строка разбивается на токены, такие как наименования точек (именные точки типа «A», «B» преобразуются в порядковые числа, которыми оперируют алгоритмы), числа, критерии, символы, отвечающие за операторы. Это позволяют структурировать данные для дальнейшего анализа;

– синтаксический анализ: используется рекурсивный спусковой обработчик для построения абстрактного синтаксического дерева (AST) на основе грамматики языка;

– семантическая проверка: на этом этапе проверяется корректность ссылок на точки и критерии, а также валидность диапазонов позиций и выражений. Например, что точка «G» существует, а диапазон «[2, 7]» имеет смысл;

– для оценки условий используется стековый подход: строка в польской нотации обрабатывается справа-налево. Операнды помещаются в стек, а операторы извлекают нужное количество операндов, выполняют операцию и возвращают результат в стек;

– AST преобразуется в объект Rule;

В нотации BNF грамматика AST определяется следующим образом [45, 46, 47]:

expression ::= condition "->" sanctions

*sanctions ::= sanction ("," sanction)**

sanction ::= criterion ":" value | "exc"

condition ::= logical_or_expression

*logical_or_expression ::= logical_and_expression ("|" logical_and_expression)**

*logical_and_expression ::= not_expression ("&" not_expression)**

not_expression ::= "!" not_expression | comparison_expression

comparison_expression ::= value comparison_operator value | position_expression | order_expression | function_call

value ::= number | function_call

function_call ::= "@" city city criterion

criterion ::= word

comparison_operator ::= ">" | "<" | "==" | "!=" | etc.

position_expression ::= "#" city "~" range | "#" city "=" number

range ::= "[" [number] "," [number] "]" | "[" [number] "," [number] ")"

order_expression ::= ">>" city city | ">>>" city city

city ::= [A-Z]

number ::= [0-9]+

word ::= [a-z]+

Интерпретатор является программным модулем, который будет принимать на вход текст на языке ограничений, а возвращать объект Rule, содержащий в себе функцию для проверки и словарь, содержащий значения по критериям для суммирования (санкции) или метку об исключении.

3.5 Программная реализация эволюционных алгоритмов

В таблице 3.2 приводится описание типов в среде Rust, которые использовались в реализациях и упоминались в схемах алгоритмов.

Таблица 3.2 Описание интегрированных Rust-типов

Наименование	Описание
usize	Беззнаковые целые числа
f32	Числа с плавающей точкой (32 бит)
f64	Числа с плавающей точкой (64 бит)
Option	Содержит значение или его отсутствие (None)
Result	Аналог Option, но содержит значение или объект ошибки
FxHashMap (HashMap)	Словарь типа ключ-значение
Vec	Упорядоченная последовательность значений динамической длины (вектор)
String	Расширенный строковый тип
(…, …)	Коллекция значений фиксированной длины (кортеж)
ThreadRng	Генератор случайных последовательностей
Enum	Перечисление
bool	Булево (логическое) значение
Ordering	Представление результата сравнения двух значений

Общая диаграмма классов показана на рисунке 3.5.

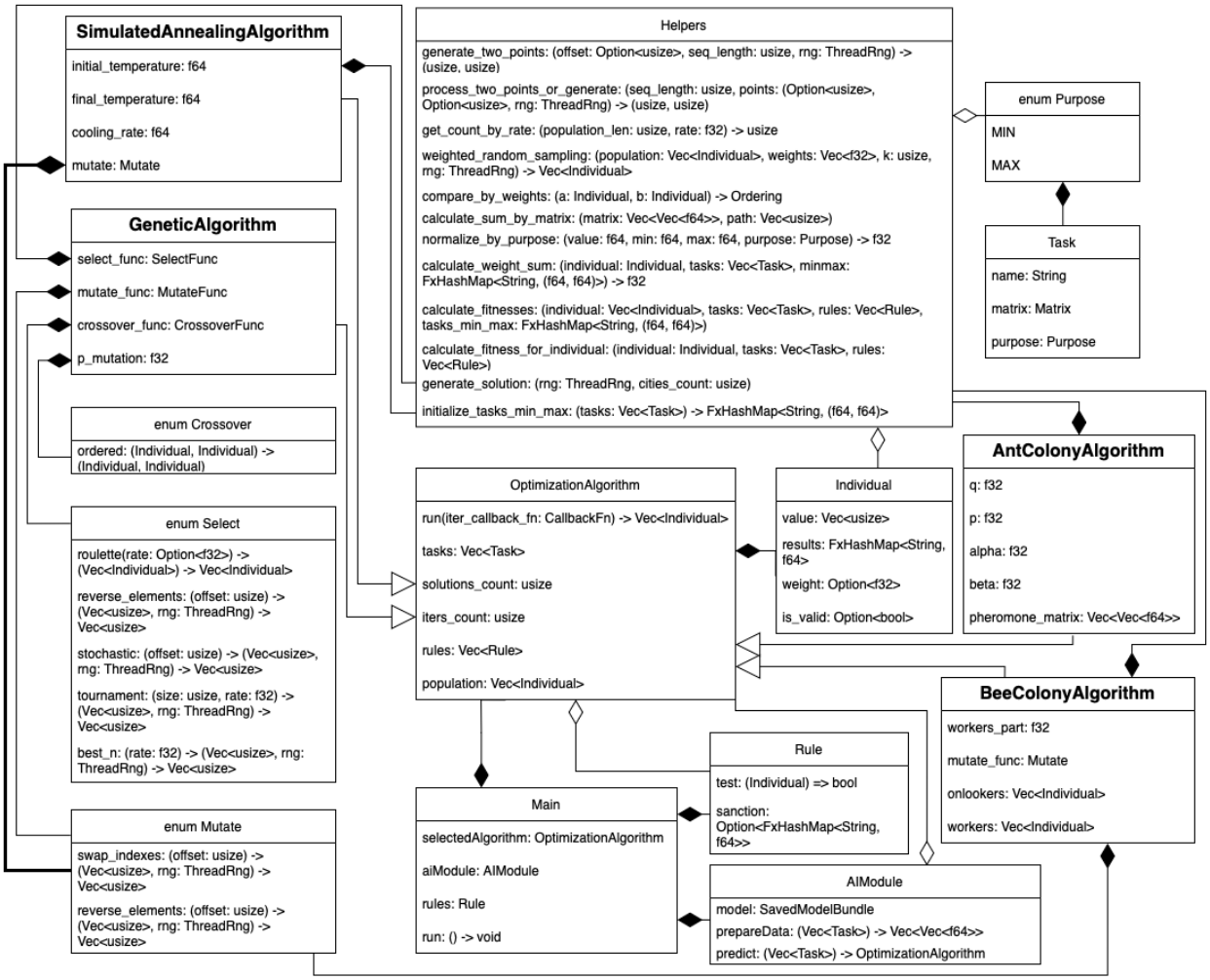


Рисунок 3.5 – Диаграмма классов

Тремя основными классами в системе, которые используются в алгоритмах, являются типа Individual, Task, Rule.

Класс Individual отвечает за работу акторов. Он содержит следующие поля:

– value - последовательность точек, являющаяся предлагаемым решением (x);

– results отвечает за хранение приспособленности решения по критериям, вычисленной по формуле 2.5;

– weight: МКС. Имеет значение None если еще не вычислено или решение подлежит исключению;

– is_valid является булевым значением, которое показывает, удовлетворяет ли решение ограничениям или нет. Имеет значение None если еще не вычислено. На основе данного атрибута производится фильтрация акторов;

Кроме того, класс Individual выполняет роль выходных данных. После завершения работы алгоритма, он возвращает массив экземпляров Individual с подробным описанием решений. Данный тип данных можно представить в формате JSON-строки или обратно из JSON в «программный» вид, что значительно упрощает интеграцию в другие системы.

Класс Task является классом, представляющим отдельный критерий. Он хранит всю ту информацию о критериях, которая требуется алгоритмам идентификации и расчетов по ним. Имеет следующие атрибуты:

– name является строковым атрибутом, содержащим наименование критерия. Используется алгоритмами для корректного вычисления МКС, а сущностью Rule – для валидации акторов и применение к ним санкций;

– matrix хранит матрицу смежности;

– purpose – цель критерия (MIN или MAX).

Класс Rule отвечает за ограничения. Является результатом работы интерпретатора. Содержит метод test для проверки решения на предмет удовлетворения ограничений и атрибут sanction, в котором описаны санкции для каждого из критериев по имени. Если sanction имеет значение None невалидное решение подлежит исключению. Неудовлетворение ограничению подразумевает возврат значения true из функции test.

Класс OptimizationAlgorithm является родительским классом для классов эволюционных алгоритмов. Он содержит все поля, которые используются между алгоритмами. Таким образом, для каждого эволюционного алгоритма имеются следующие обязательные поля:

– tasks является массивом, состоящим из экземпляров Task. Содержит всю информацию о критериях. Подается аргументом на вход конструктору класса;

– rules – массив из Rule. Содержит всю информацию об ограничениях. Подается аргументом на вход конструктору класса;

– solutions_count является целочисленным параметром, сообщающим, сколько решений должен вывести алгоритм. Каждый из алгоритмов формирует массив лучших решений длиной в solutions_count в конце каждого итерационного шага. Это было сделано для контроля работы алгоритма и его остановки.

– iters_count также является целочисленным параметром, который отвечает за количество итерационных шагов, которое должен проделать алгоритм в своей работе;

– population представляет собой массив акторов, который изменяется с каждым итерационным шагом. Может подаваться на вход алгоритму для восстановления контекста.

Класс AIModule отвечает за ИИ-составляющую. В нем открывается и хранится ИИ-модель, инициализируется база знаний. Также в нем обрабатываются данные для подачи ИИ-модели, и формируется интеллектуальное решение (прогноз) по дальнейшему решению задачи. В структурной схеме его представляет Модуль принятия решений.

В классе Helpers содержатся все общие методы, используемые эволюционными алгоритмами.

Метод generate_two_points используется методами мутации для корректной и безопасной генерации двух случайных точек. Данный метод устроен так, что возвращает функцию с работающим замыканием и уже указанным сдвигом (разницей) между точек. Сгенерированная функция выступает как аргумент для некоторых алгоритмов, использующих мутацию. Это необходимо для того, чтобы сделать функцию заранее готовой к вызову без передачи дополнительных аргументов и вызвать ее в совершенно другом месте и с другим контекстом, а именно, внутри итерационного цикла алгоритма. Функция возвращает кортеж, состояний из двух точек. Схема алгоритма функции, возвращаемой generate_two_points показана на рисунке 3.6.

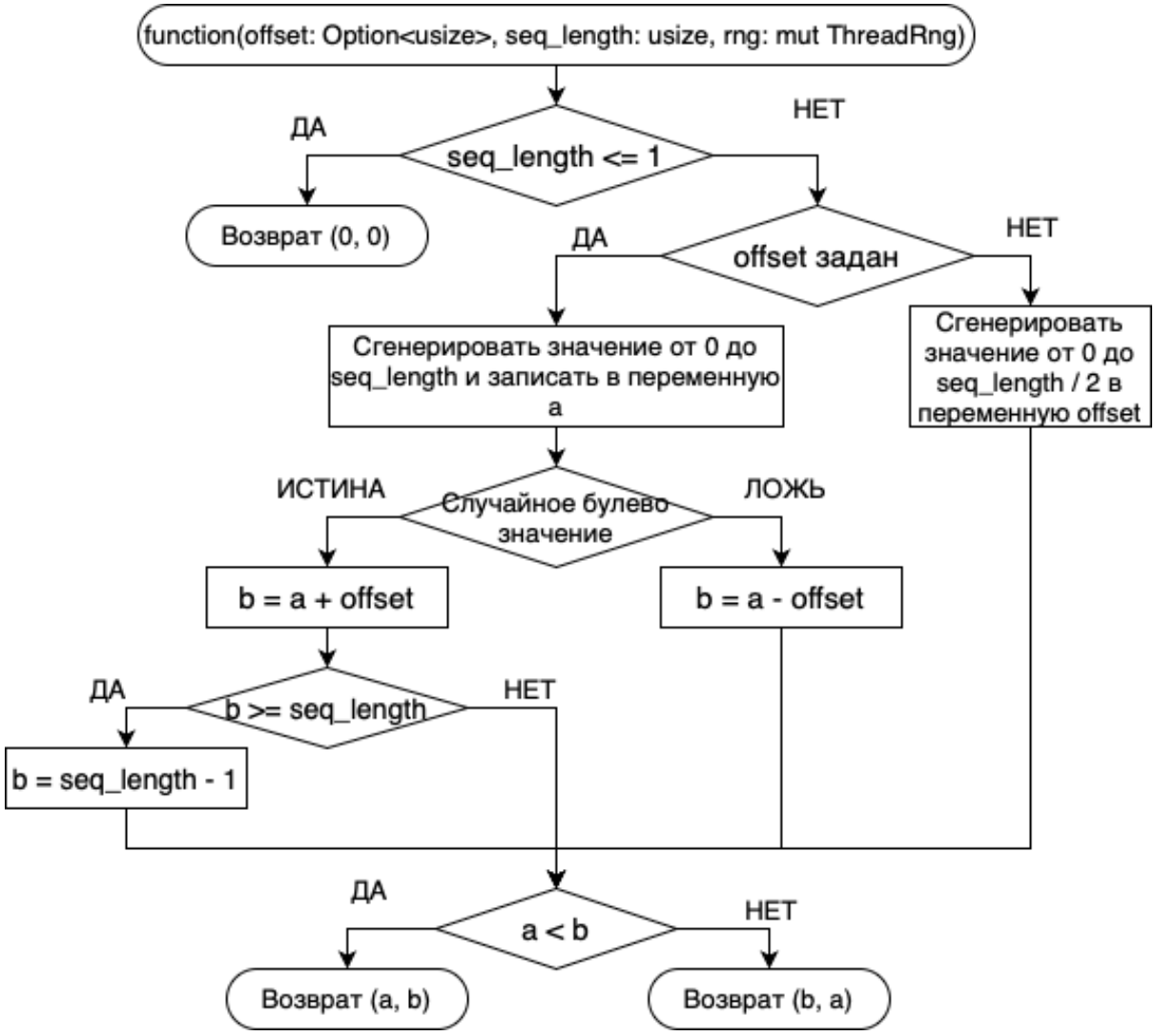


Рисунок 3.6 – Схема алгоритма generate_two_points

Из основных операций в данной функции, это генерация первой точки, затем генерация второй точки, являющаяся суммой заданного (или сгенерированного) сдвига и первой точки, коррекция второй точки и возврат в порядке возрастания. Это решение полностью предотвращает появление различных ошибок в вычислительном процессе алгоритмов.

Метод get_count_by_rate используется, в основном в методах отбора ГА, где известен процент особей, который должен остаться, но неизвестно их общее количество. Данный метод вычисляет количество на основе численности множества и множителя, как показано в схеме на рисунке 3.7.

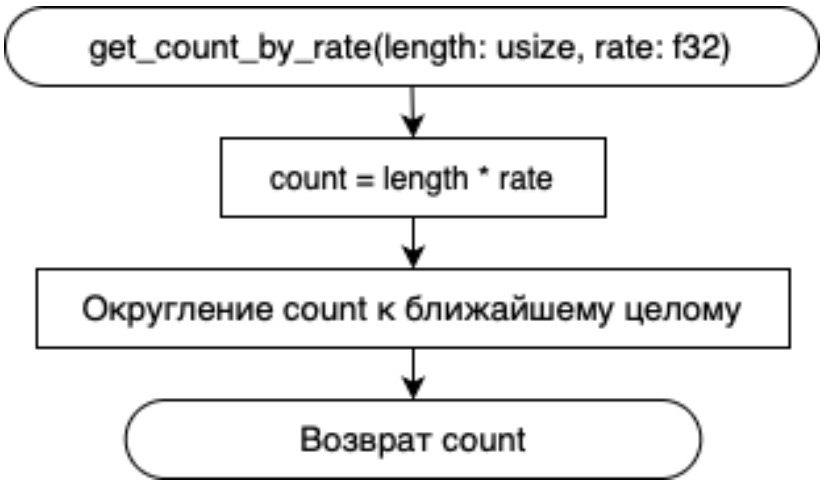


Рисунок 3.7 – Схема алгоритма get_count_by_rate

Метод weighted_random_sampling широко используется в ГА и ПА и позволяет получить случайное значение из множества на основе предоставленных весов. Данный метод получает множество целевых значений и множество с весами, где каждому весу соответствует значение из первого множества. Другими словами, данный метод представляет вероятностную функцию. Кроме того, метод способен возвращать более чем одно значение. Схема алгоритма weighted_random_sampling показана на рисунке 3.8.

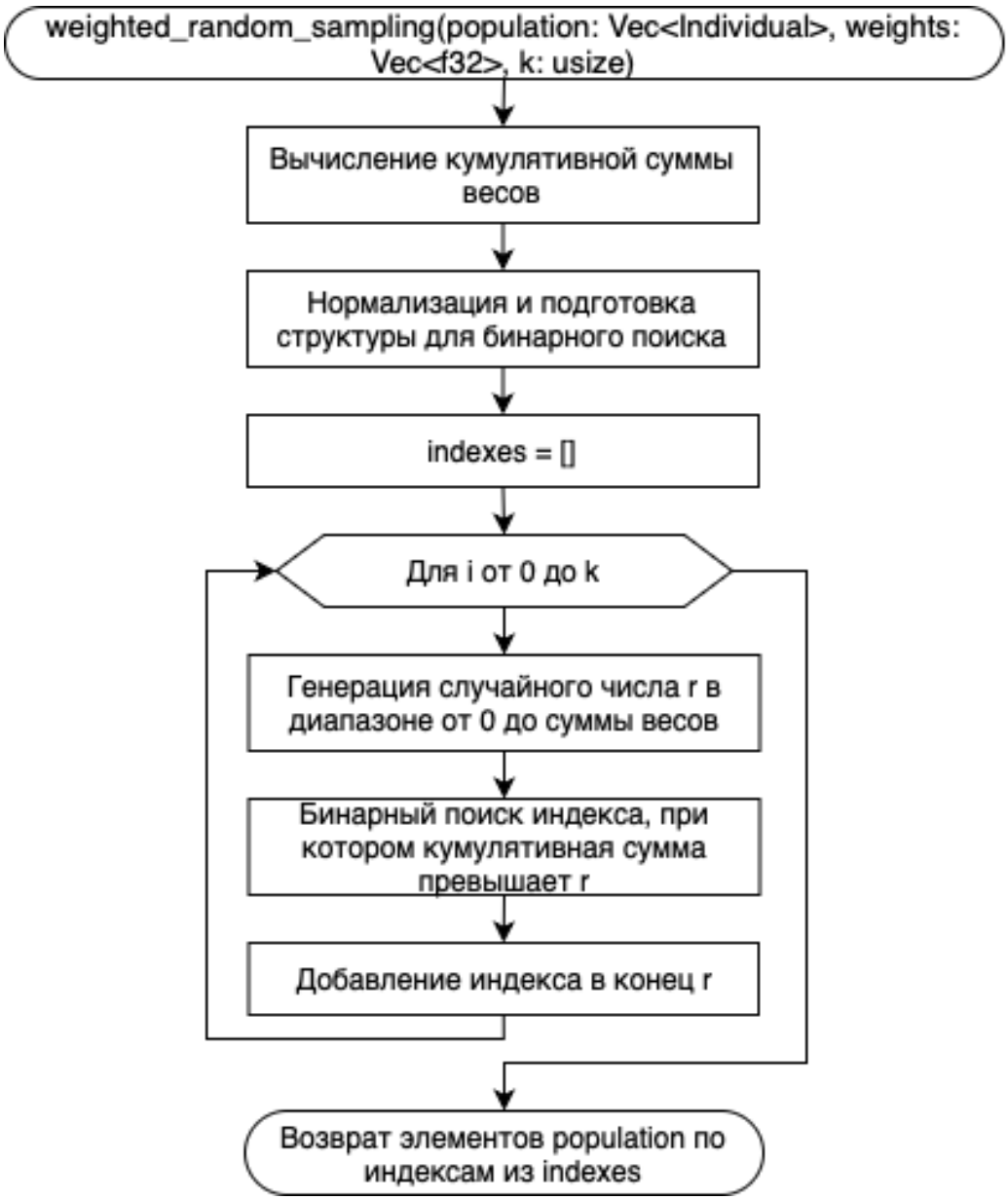


Рисунок 3.8 – Схема алгоритма weighted_random_sampling

Алгоритм weighted_random_sampling выполняет выборку k элементов из популяции population, используя взвешенное случайное распределение на основе заданных weights. Сначала вычисляется кумулятивная сумма весов, что позволяет преобразовать веса в интервалы вероятностей. Затем выполняется нормализация и подготовка структуры данных для эффективного бинарного поиска.

После этого создается пустой список indexes, и начинается цикл, выполняющий k итераций. В каждой итерации генерируется случайное число r в диапазоне от 0 до суммы всех весов. Затем с помощью бинарного поиска

определяется индекс первого элемента, у которого кумулятивная сумма превышает r. Этот индекс добавляется в список indexes.

По завершении цикла элементы из population выбираются по индексам из indexes, после чего алгоритм возвращает результат. Такой метод позволяет делать выборку элементов пропорционально их весам, что полезно, например, в эволюционных алгоритмах или при отборе вероятностных решений.

Метод compare_by_weights используется во всех алгоритмах. Он является сравнительным методом и используется в качестве аргумента для функций сортировки. В данном случае, используется для сортировки акторов по их МКС. В связи с особенностью языка Rust, данный метод возвращает объект Ordering, который имеет одно из следующих значений: Less (меньше), Greater (больше), Equal (равно). Схема алгоритма метода compare_by_weights показана на рисунке 3.9.

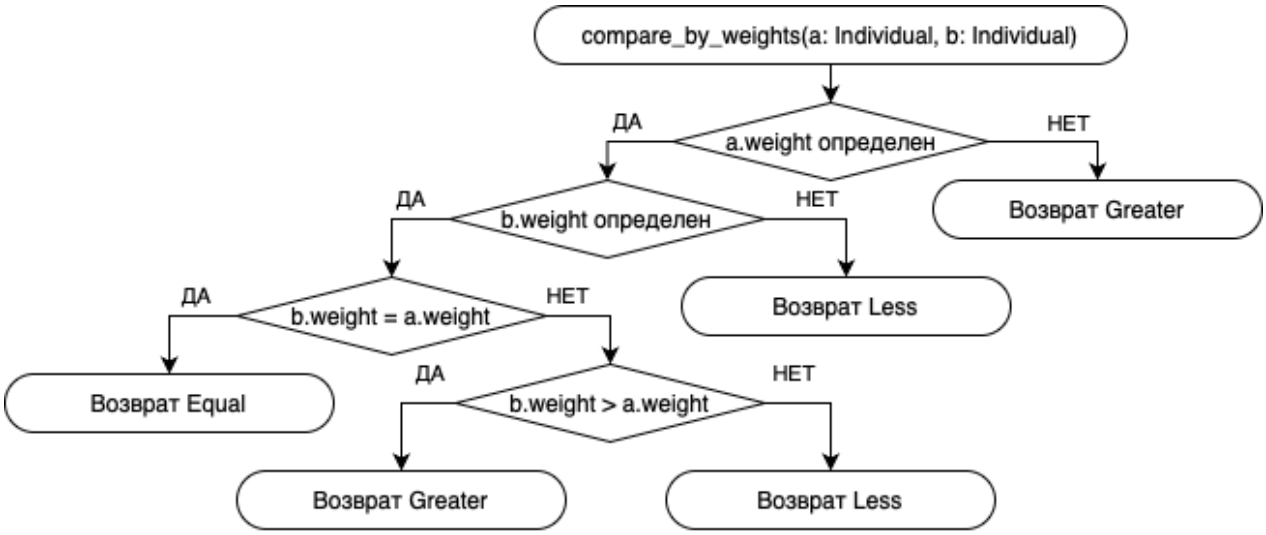


Рисунок 3.9 – Схема алгоритма compare_by_weights

Принцип работы данного метода заключается в сравнении МКС актора A и актора B. Тот актор, у которого МКС больше, оказывается в первых местах сортируемого множества. Акторы с отсутствующими МКС безусловно перемещаются в конец множества и, в дальнейшем, удаляются из итогового решения.

Стоит упомянуть, что для сортировки значений Rust использует гибридный алгоритм сортировки [78, 79]. В первую очередь, он использует быстрый способ сортировки. Если в процессе вычислений Rust сталкивается с «плохим разбиением» (т.е. глубина рекурсии превышает ln(n), где n – численность множества), он переходит на пирамидальный способ сортировки. На подмножествах, Rust использует метод вставками [80, 81].

Метод calculate_sum_by_matrix используется в каждом эволюционном алгоритме для того, чтобы вычислять сумму условных расстояний на основе представленной матрицы смежности и решения (последовательности точек). Схема алгоритма показана на рисунке 3.10.

Сначала инициализируется переменная sum, накапливающая сумму значений. Затем начинается цикл, который проходит по всем последовательным парам городов в path. Для каждой такой пары from и to значение из matrix[from][to] добавляется в sum.

После обработки всех пар происходит замыкание маршрута: к sum добавляется значение из matrix[first_city][last_city], где first_city — первый город в маршруте, а last_city — последний.

В конце алгоритм возвращает вычисленное суммарное значение. Такой подход позволяет оценить общий "вес" пути, например, в задачах коммивояжёра или маршрутизации.

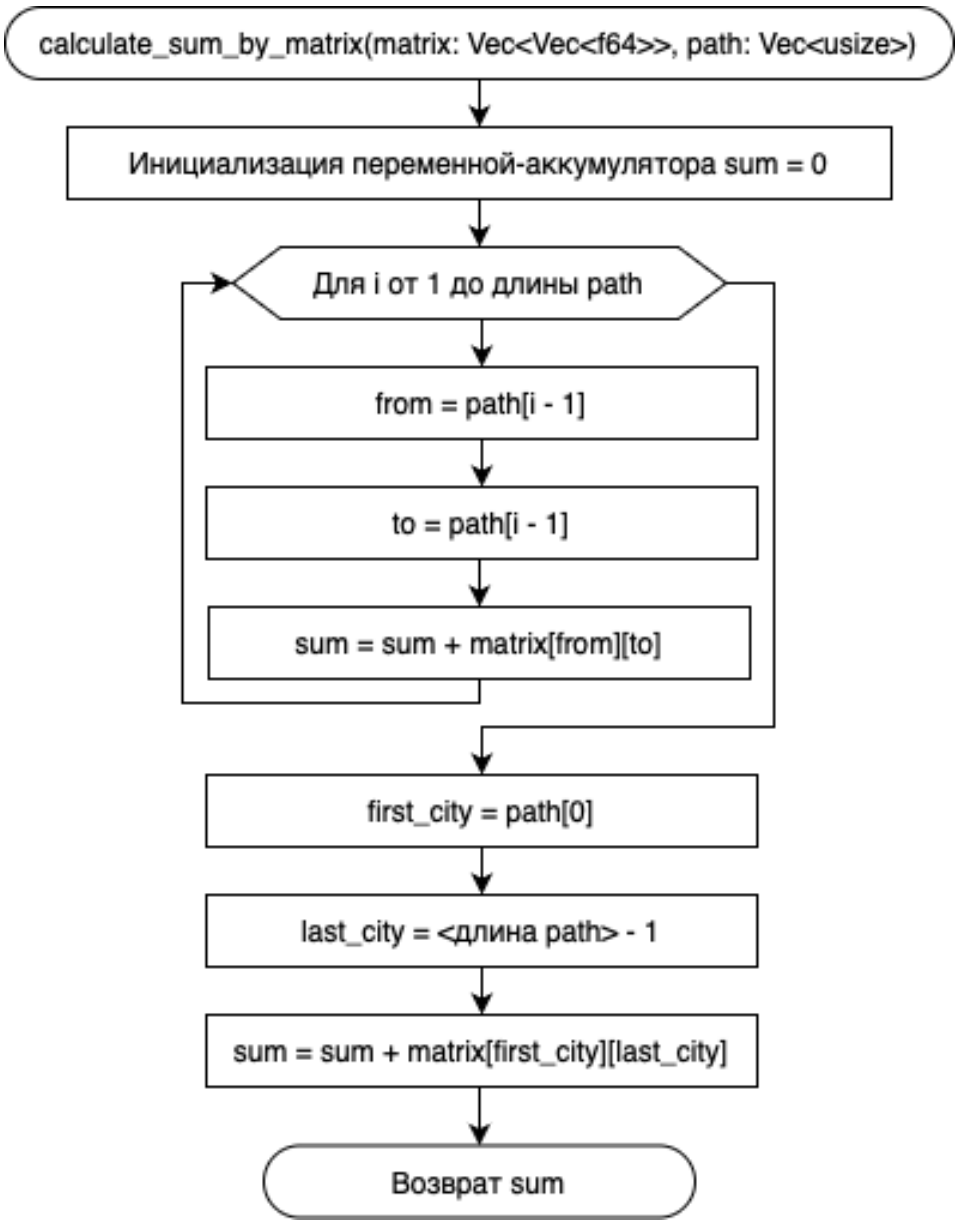


Рисунок 3.10 – Схема алгоритма calculate_sum_by_matrix

Метод normalize_by_purpose отвечает за вычисление МКС в зависимости от наибольшего значения по критерию, наименьшего, цели критерия и текущего значения. Данный метод широко во всех эволюционных алгоритмах. Схема алгоритма показана на рисунке 3.11. Стоит отметить, что начальная проверка, где вычисляется разница между max и min нужна для того, чтобы в знаменателе формулы в дальнейшем не оказалось крайне малого значения или нуля. Если разница между max и min близка к нулю, ничего не остается, кроме как присвоить решение значение единицы (максимальное) по данному критерию.

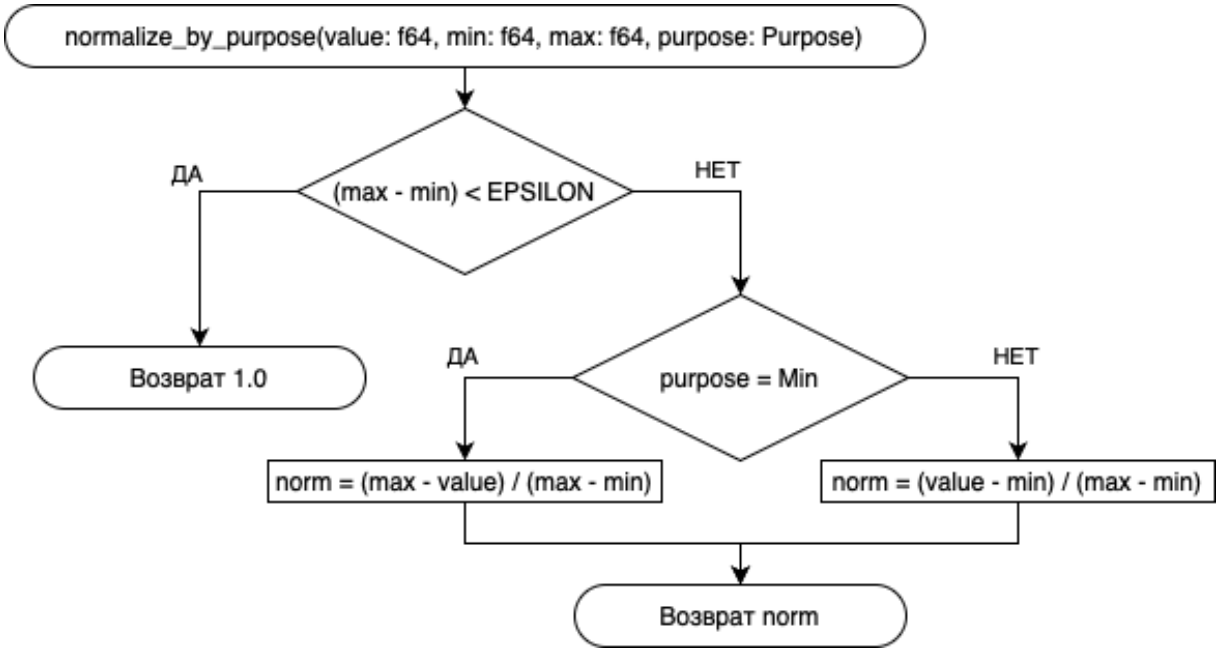


Рисунок 3.11 – Схема алгоритма normalize_by_purpose

Для вычисления МКС и сразу их суммы, что является конечной сравнительной характеристикой, используется метод calculate_weight_sum. Необходимо, чтобы результаты приспособленности по критериям уже были подсчитаны, и определены минимальные и максимальные значения приспособленностей на текущие популяцию и итерационный шаг. Данный метод «пробегается» по каждому существующему в задаче критерию, вычисляет для каждого из них МКС (путем вызова normalize_by_purpose) и суммирует результаты. Схема алгоритма показана на рисунке 3.12.

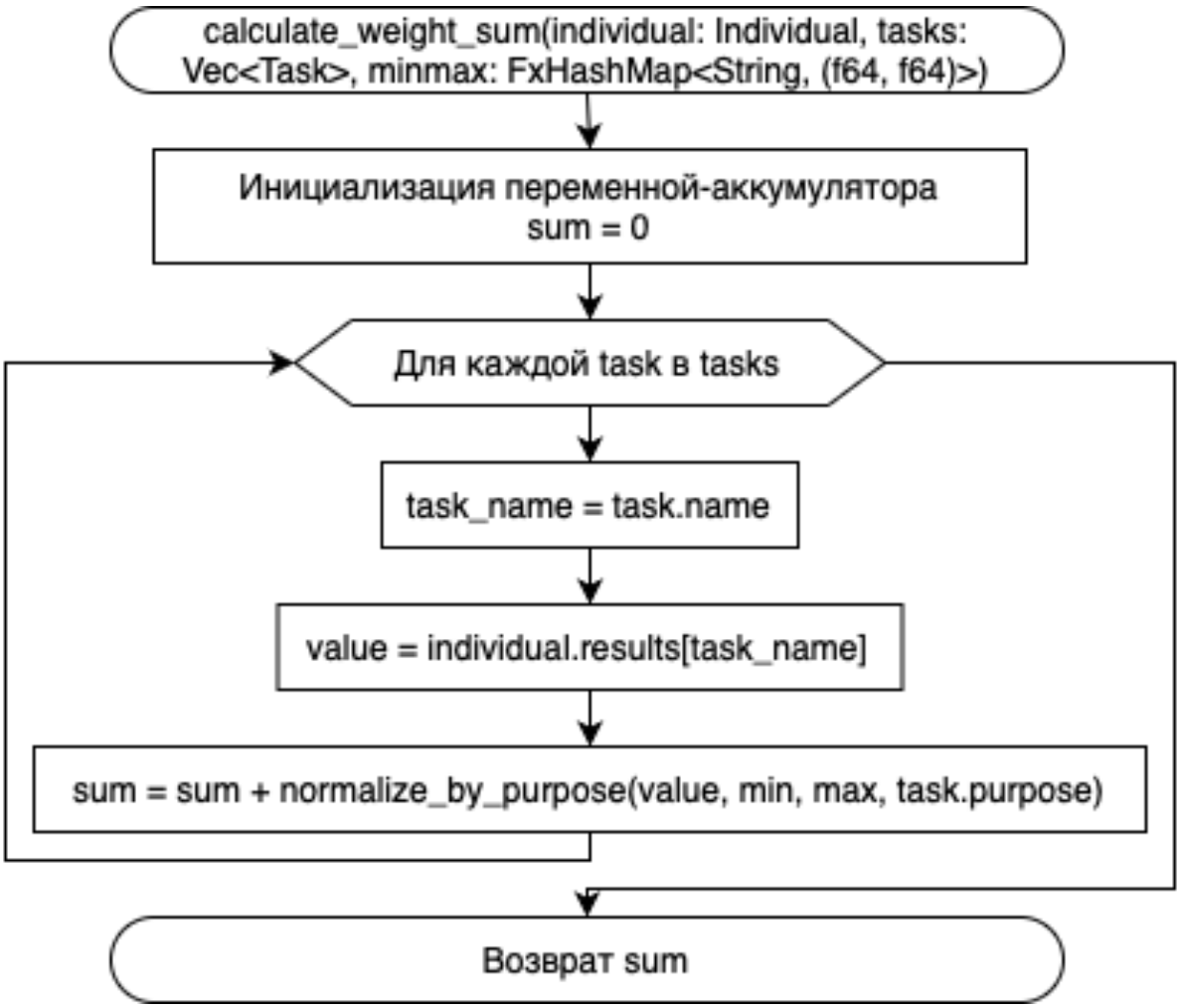


Рисунок 3.12 – Схема алгоритма calculate_weight_sum

Далее будут описаны наиболее крупные функции системы. Одной из таких функций является функция вычисления приспособленности актора по критериям с учетом ограничений под названием calculate_fitness_for_individual, принимающая на вход объект актора, заданные критерии (Task), и заданные ограничения (Rule). Схема алгоритма приведена на рисунке 3.13.

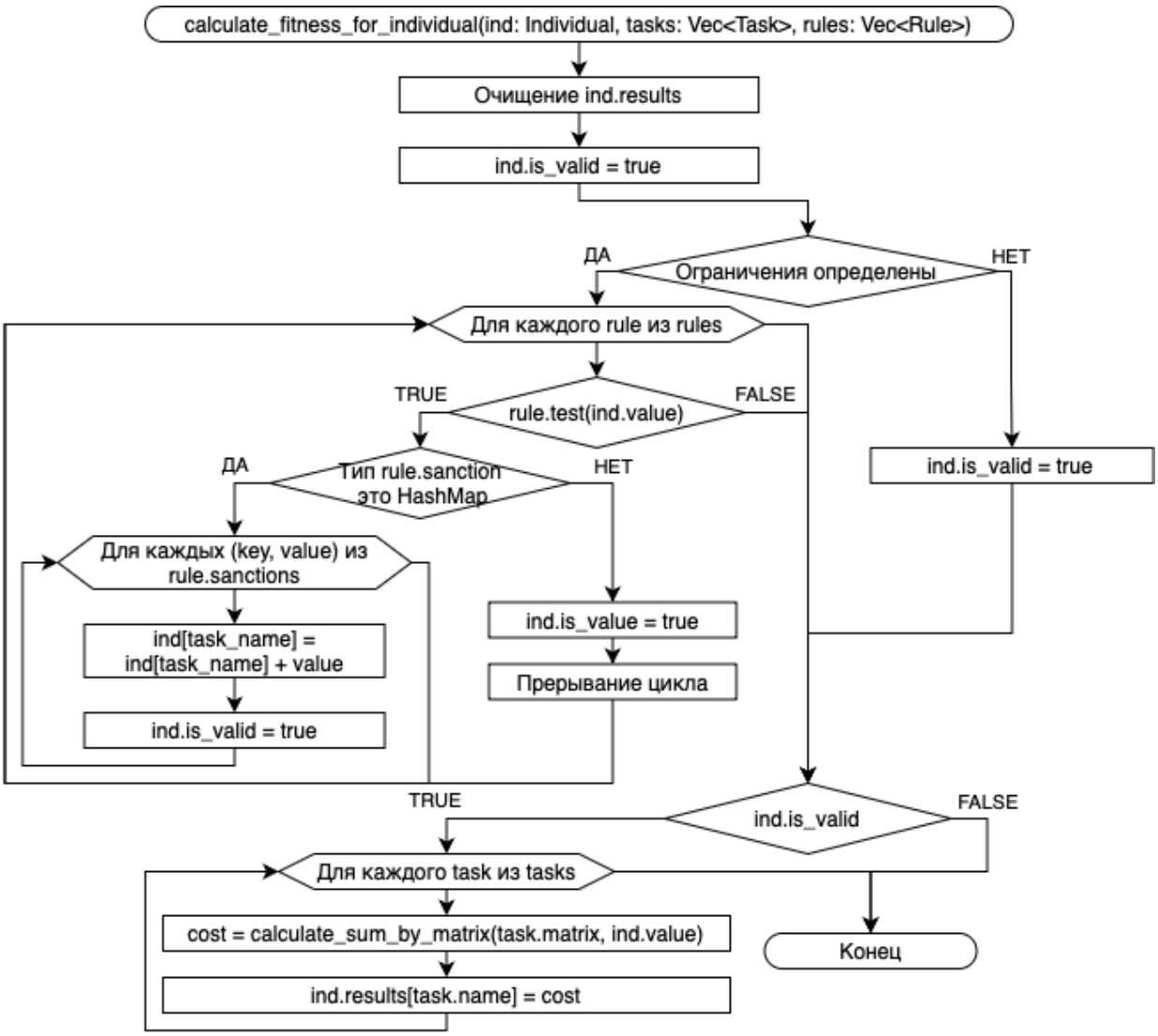


Рисунок 3.13 – Схема алгоритма calculate_fitness_for_individual

В начале очищаются существующие результаты у актора, и устанавливается флаг ind.is_valid в true. Далее проверяется, заданы ли ограничения. Если они отсутствуют, ind.is_valid остается true, и работа метод пропускает описанный далее шаг.

Если ограничения есть, начинается обработка правил. Для каждого правила выполняется тест rule.test(ind.value). Если тест успешен, проверяется тип rule.sanction. Если это HashMap, то перебираются все пары (key, value), и к соответствующему значению в ind прибавляется value. При этом ind.is_valid остается true.

Если же тест правила не пройден, устанавливается флаг ind.is_valid устанавливается в false, и выполнение правил прекращается. После обработки

правил проверяется, является ли ind допустимым. Если нет, выполнение функции завершается досрочно. Если да, то для каждой задачи вычисляется cost с помощью функции calculate_sum_by_matrix, и результат сохраняется в ind.results по ключу, соответствующему критерию, по которому велось вычисление. На этом алгоритм завершается.

Для вычисления МКС для всей популяции акторов был реализован метод calculate_fitnesses (рисунок 3.14).

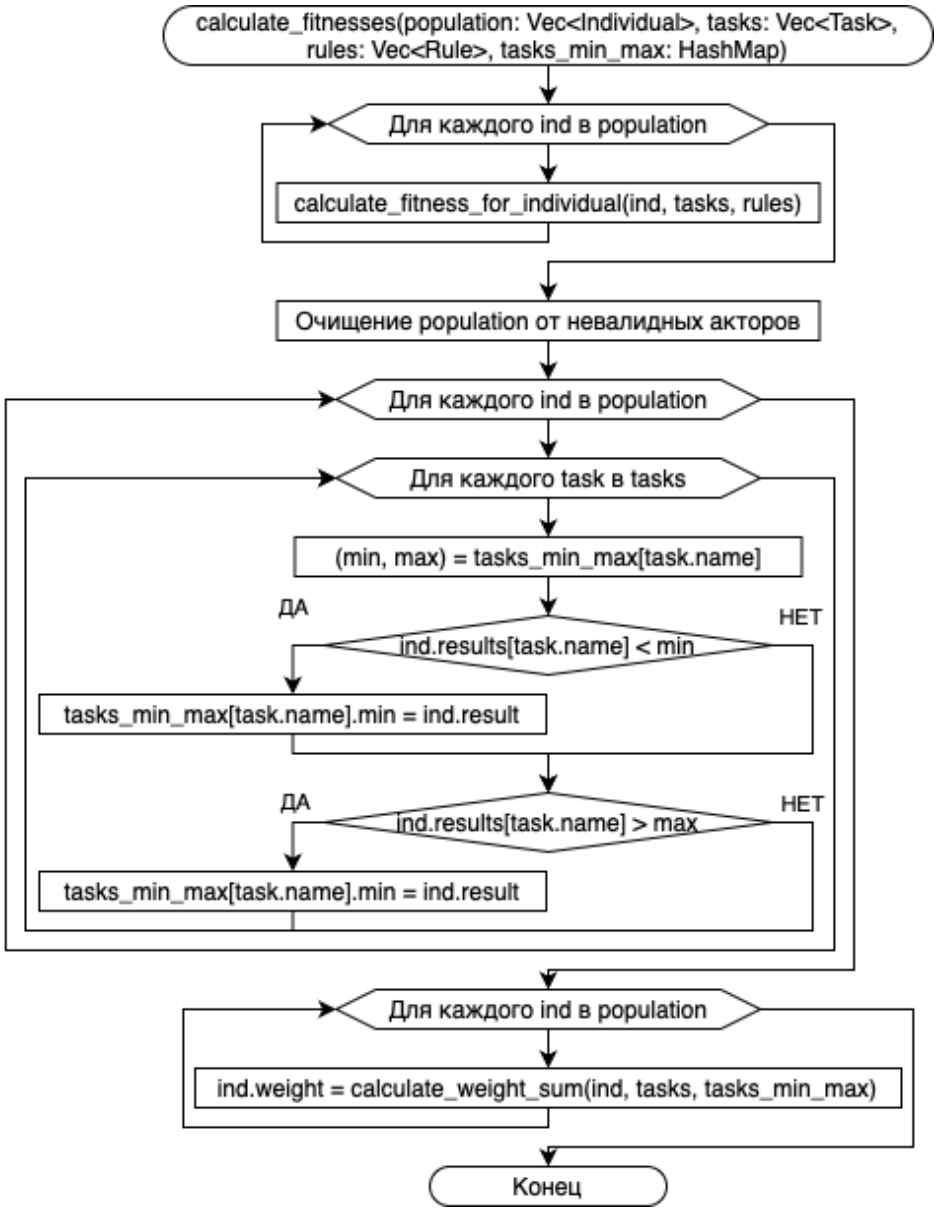


Рисунок 3.14 – Схема алгоритма calculate_fitnesses

В данном методе для каждого элемента вызывается функция calculate_fitness_for_individual, которая оценивает приспособленность на

основе списка критериев tasks и набора ограничений rules. После этого из популяции удаляются особи, подлежащие исключению по причине неудовлетворения ограничениям.

Затем снова перебираются все элементы популяции. Для каждого актора и каждого критерия определяются минимальные и максимальные значения, которые в дальнейшем записываются переменную tasks_min_max. Если приспособленность актора по критерию меньше текущего минимума, учет минимального результата обновляется. Если результат больше текущего максимума, соответствующим образом обновляется учет максимального результата.

На последнем этапе для каждого элемента популяции вычисляется МКС с помощью описанной ранее функции calculate_weight_sum, которая использует обновленные tasks_min_max.

Метод генерации решения (generate_solution) активно используется в ГА, ПА и ИО. Для его работы необходимо подать на вход только численность точек в задачах. Схема алгоритма данного метода приведена на рисунке 3.15.

Сначала инициализируется пустое множество solution. Затем в цикле от 0 до cities_count - 1 в этот список поочередно добавляются числа от 0 до cities_count - 1, формируя упорядоченную последовательность.

После этого выполняется перемешивание элементов. В новом цикле (от 0 до cities_count - 1) выбирается случайный индекс j в диапазоне от нуля включительно до i включительно. Затем происходит обмен значений solution по индексу i и solution по индексу j, что приводит к случайному распределению элементов списка.

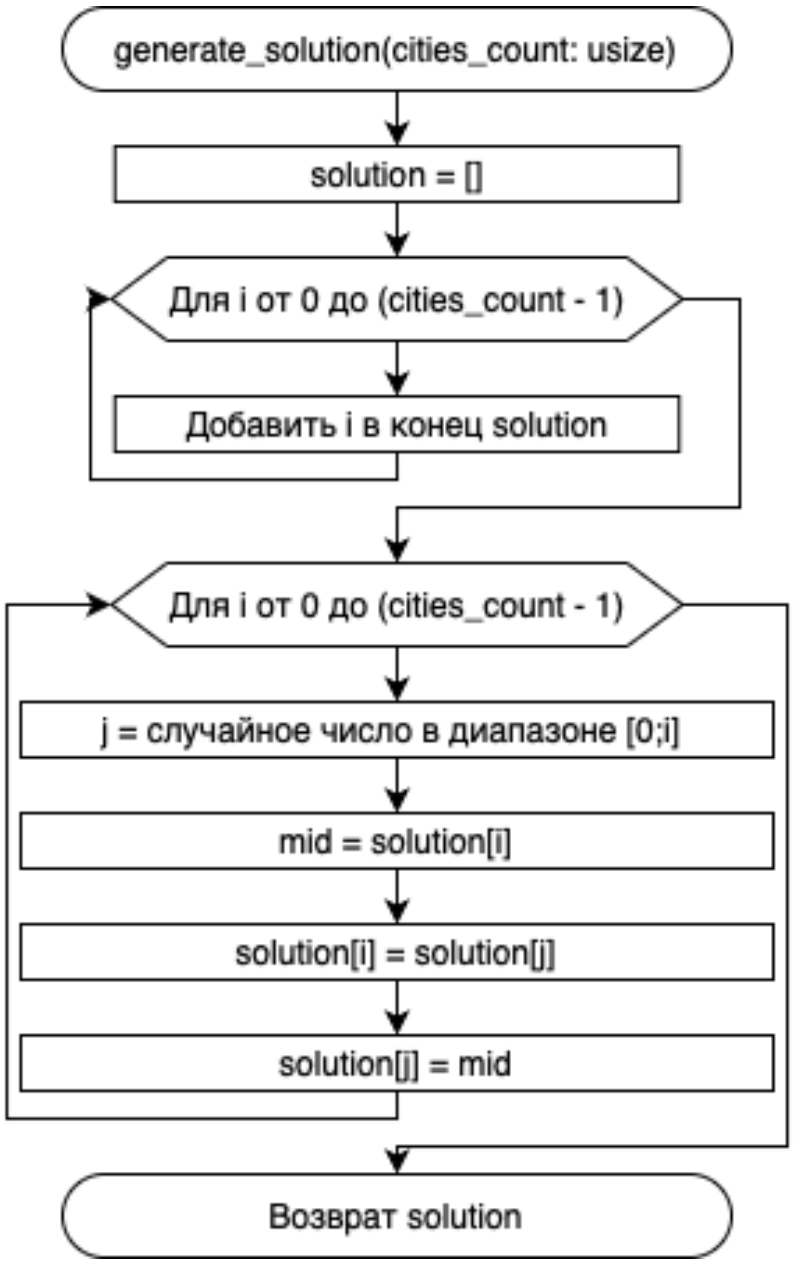


Рисунок 3.15 – Схема алгоритма generate_solution

Как было описано ранее, информацию о минимальном и максимальном значениях приспособленности между акторами внутри итерации хранится в экземпляре класса, отвечающего за алгоритм в переменной tasks_min_max, имеющая структуру «ключ-значение», в котором ключом является наименование критерия, значением – кортеж из минимального и максимального значений. Эти значения записываются, например при работе функции calculate_fitnesses. Для простой и «системной» генерации tasks_min_max используется функция initialize_tasks_min_max (рисунок 3.16).

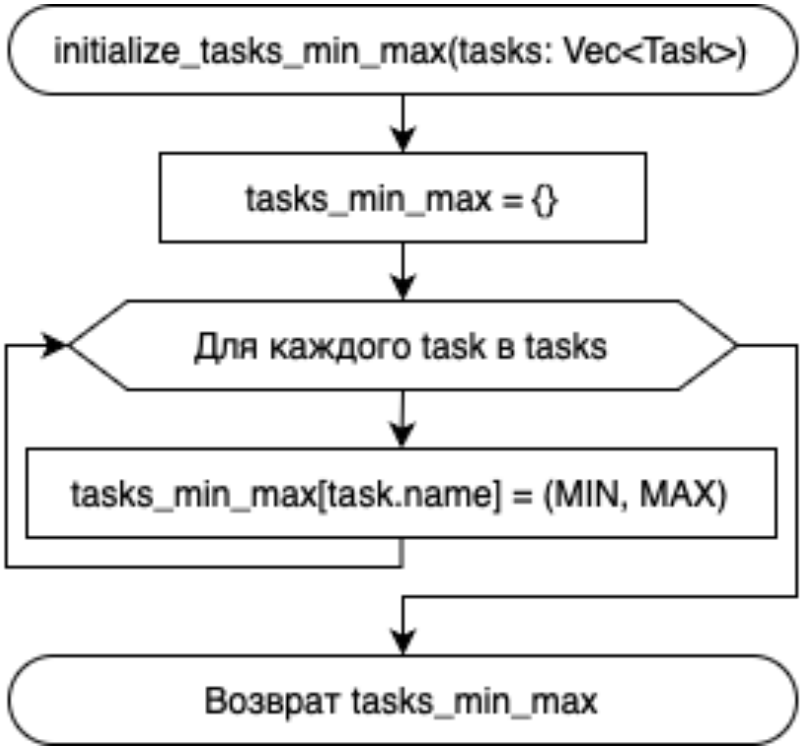


Рисунок 3.16 – Схема алгоритма initialize_tasks_min_max

Сначала создается пустая структура tasks_min_max. Затем для каждого критерия task в tasks в эту структуру добавляется запись, где ключом является имя задачи, а значением — кортеж из MIN и MAX, обозначающий начальные граничные значения. Под MIN и MAX здесь подразумеваются глобальные переменные, которые хранят в себе минимально и максимально возможные вещественные числа. После завершения инициализации структура tasks_min_max возвращается в качестве результата.

Схема метода run MA показана на рисунке 3.17.

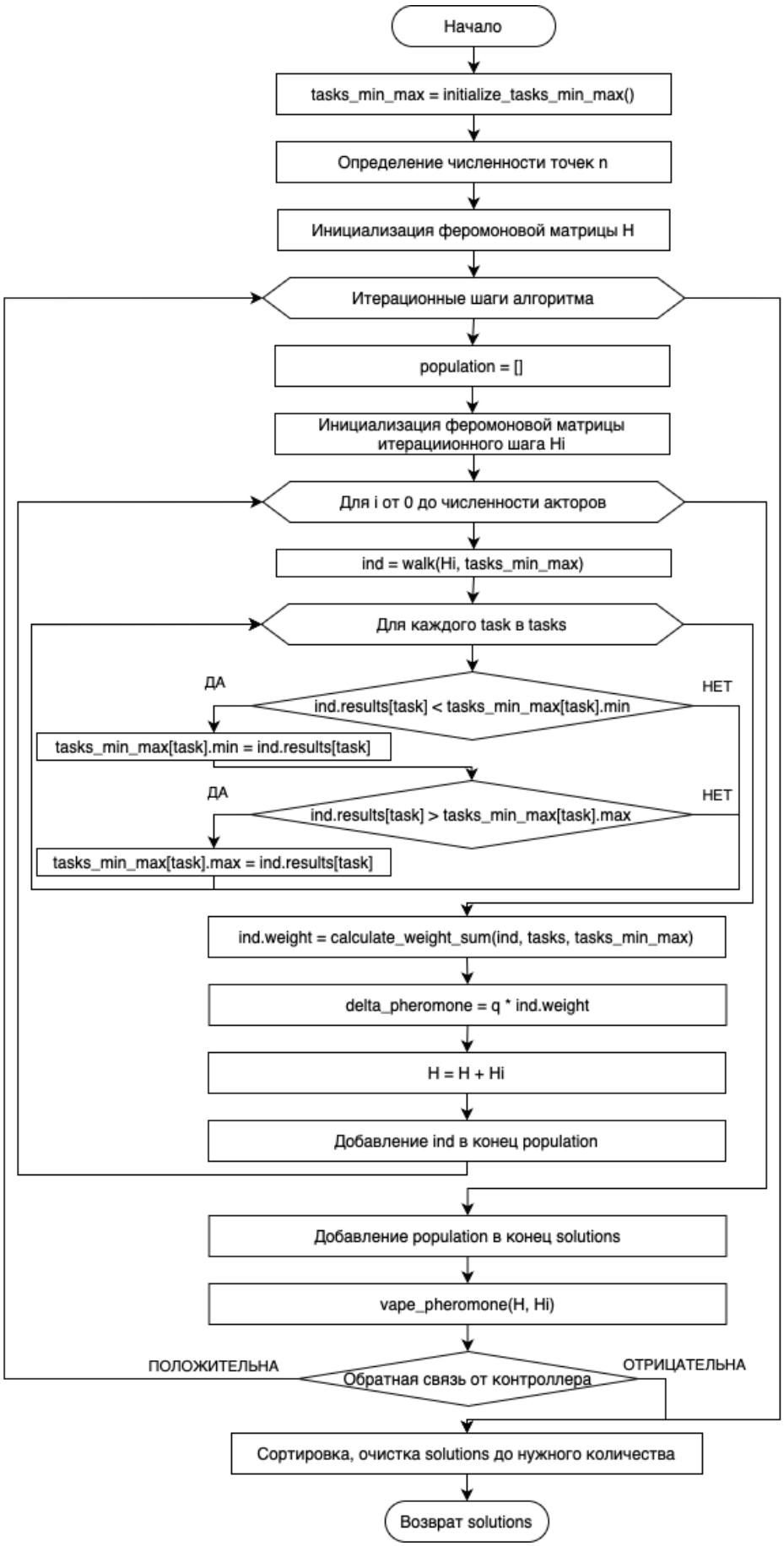


Рисунок 3.17 – Схема метода run муравьиного алгоритма

Алгоритм начинается с получения входных параметров, которые включают данные о задачах и ограничениях. Затем выполняется инициализация структуры tasks_min_max, которая хранит минимальные и максимальные значения для каждой задачи. После этого определяется количество точек (агентов), участвующих в процессе оптимизации, и создаётся начальная феромоновая матрица H, которая используется для управления перемещением муравьев (акторов).

Далее алгоритм переходит в основной цикл итераций. В начале каждой итерации создаётся пустая популяция решений. Затем инициализируется временная феромоновая матрица Hi, используемая на данном шаге. После этого начинается генерация решений: для каждого агента выполняется процедура walk, которая формирует маршрут на основе феромонов и ограничений.

После построения маршрута анализируются полученные результаты. Для каждой задачи проверяется, выходит ли значение за текущие границы tasks_min_max. Если найденное значение меньше минимального порога, оно обновляется. Аналогично, если значение превышает максимальный порог, оно также корректируется.

Далее вычисляется вес решения с помощью описанной ранее функции calculate_weight_sum, которая, определяет его качество на основе полученных результатов. Затем рассчитывается приращение феромона delta_pheromone, которое зависит от веса решения. Феромоновая матрица обновляется, суммируя Hi и H. После этого полученное решение добавляется в популяцию.

После обработки всех агентов вся популяция сохраняется в общий список решений solutions. Затем выполняется процедура vape_pheromone, отвечающая за испарение феромонов, чтобы предотвратить их чрезмерное накопление и способствовать поиску новых решений.

В завершение итерации производится сортировка и очистка решений, чтобы оставить только необходимое количество лучших вариантов. Алгоритм продолжает итерации, пока не будет достигнуто заданное условие завершения. После этого выполнение заканчивается.

Далее, будут приведены схемы алгоритмов вспомогательных методов МА. На рисунке 3.18 показан метод walk, генерирующий для актора последовательность x и вычисляющий МКС.

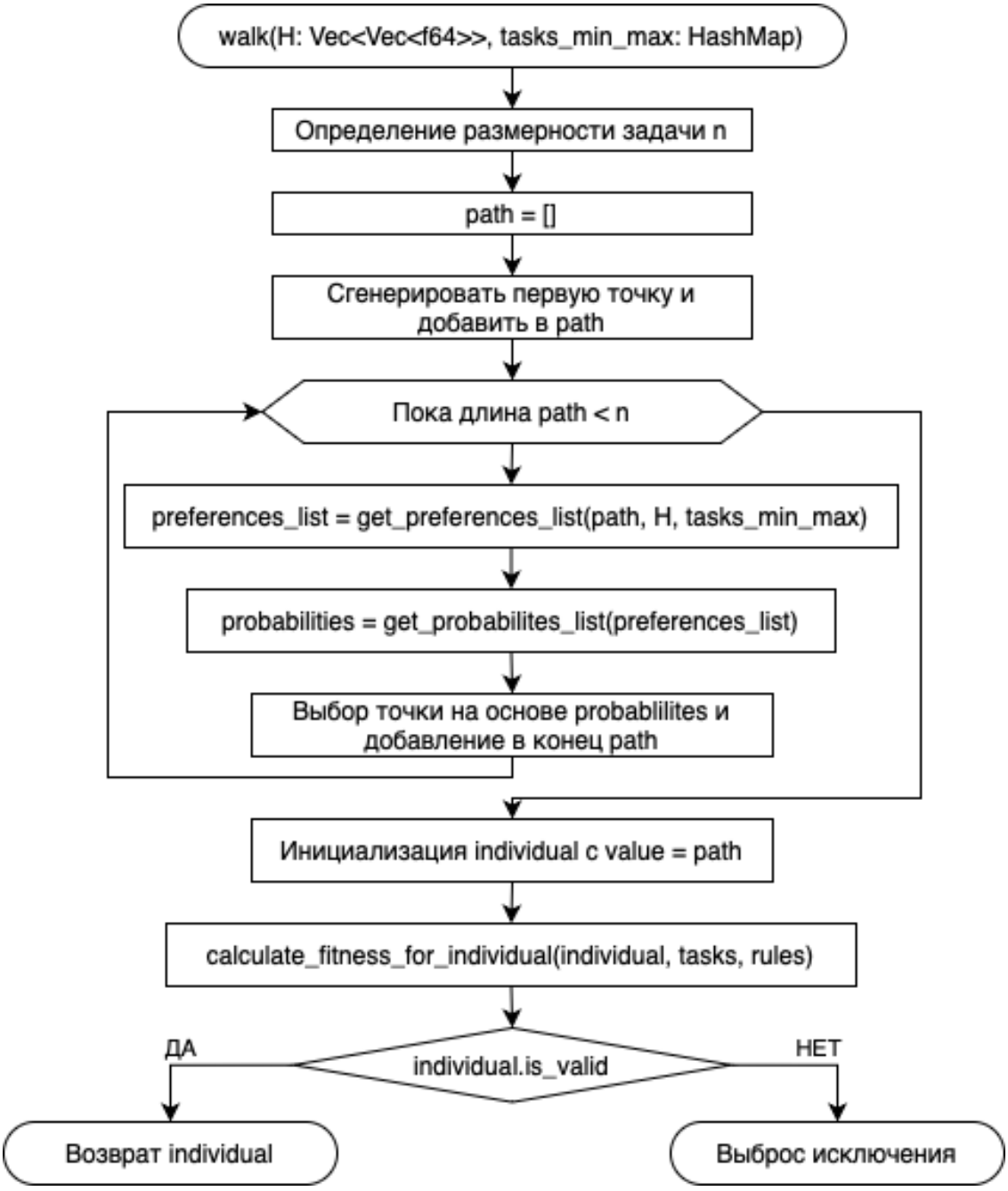


Рисунок 3.18 – Схема метода walk

Алгоритм начинается с определения размерности задачи n, то есть количества точек, которые необходимо выбрать. Затем создаётся пустой список path, в который записывается последовательность точек. Первая точка

генерируется случайным образом и добавляется в path, что служит отправной точкой для дальнейшего построения маршрута.

Далее начинается основной цикл, который продолжается, пока длина path не достигнет n. На каждом шаге:

– формируется список предпочтений preferences_list, который вычисляется на основе уже выбранного пути, феромоновой матрицы H и tasks_min_max.

– на основе предпочтений вычисляется список вероятностей probabilities, который отражает вероятность выбора той или иной точки. Из доступных точек выбирается одна случайным образом, но с учётом рассчитанных вероятностей. Выбранная точка добавляется в конец path.

– когда путь полностью построен, создаётся объект individual, которому присваивается полученный маршрут. Затем для него вычисляется значение функции приспособленности calculate_fitness_for_individual, которое оценивает, насколько данное решение соответствует задачам и установленным правилам.

На финальном этапе проверяется, является ли построенное решение допустимым (individual.is_valid). Если оно удовлетворяет всем ограничениям, individual возвращается как результат работы функции. В противном случае выбрасывается исключение, сигнализирующее о том, что построенный маршрут оказался недопустимым.

Схема алгоритма вычисления предпочтений в методе get_preferences_list показана на рисунке 3.19.

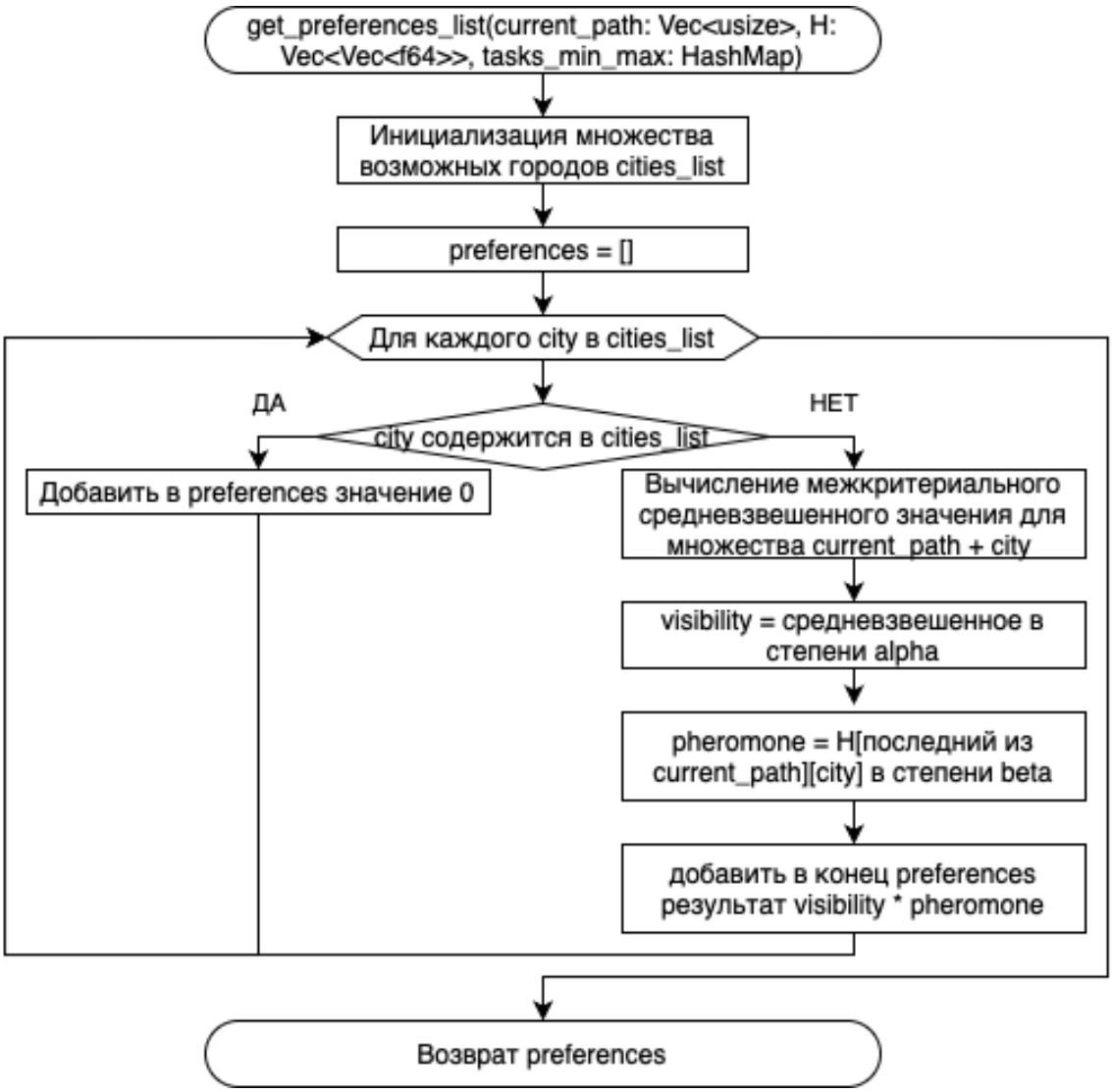


Рисунок 3.19 – Схема метода get_preferences_list

Функция get_preferences_list предназначена для вычисления предпочтений муравья при выборе следующего города на основе текущего пути current_path, феромонной матрицы H и множества ограничений tasks_min_max.

Алгоритм начинается с инициализации множества возможных точек cities_list, среди которых производится выбор. Затем создаётся список preferences, заполненный значениями, отражающими предпочтительность перехода в конкретный город.

Далее функция проходит по каждой точке city из cities_list и выполняет проверку:

– если city уже содержится в current_path, то он не должен повторяться, и в preferences добавляется нулевое значение, указывающее, что этот город не рассматривается для перехода;

– если city отсутствует в current_path, то вычисляется его предпочтительность.

Для вычисления предпочтительности рассчитывается видимость города — МКС в степени α. Далее, из феромонной матрицы H извлекается значение феромона для последнего города в current_path и рассматриваемого city, после чего это значение возводится в степень параметра β. Конечное предпочтение определяется как произведение visibility и pheromone, после чего добавляется в preferences.

Генетический алгоритм, помимо основных параметров принимает в качестве параметров методы мутации, отбора и шанс мутации. Схема метода run генетического алгоритма приведена на рисунке 3.20.

В первую очередь, генерируется начальная популяция, вычисляется ее приспособленность (в т.ч. и МКС) и инициализируется переменная tasks_min_max. Далее, идет основной итерационный цикл. В теле цикла вызывается выбранная функция отбора select_func.

После проведения отбора инициализируется переменная new_population. Для каждого актора случайным образом подбирается партнер и на основе их, с использованием метода скрещивания. В результате работы функции скрещивания образуются 2 актора-потомка в переменных child_1 и child_2, которые помещаются в new_population

Следующим шагом является мутация. Для этого, генерируется случайное вещественное число от 0 до 1. Если сгенерированное значение меньше или равно, чем p_mutation, к потомкам применяются мутации с использованием выбранной функции мутации (mutate_func).

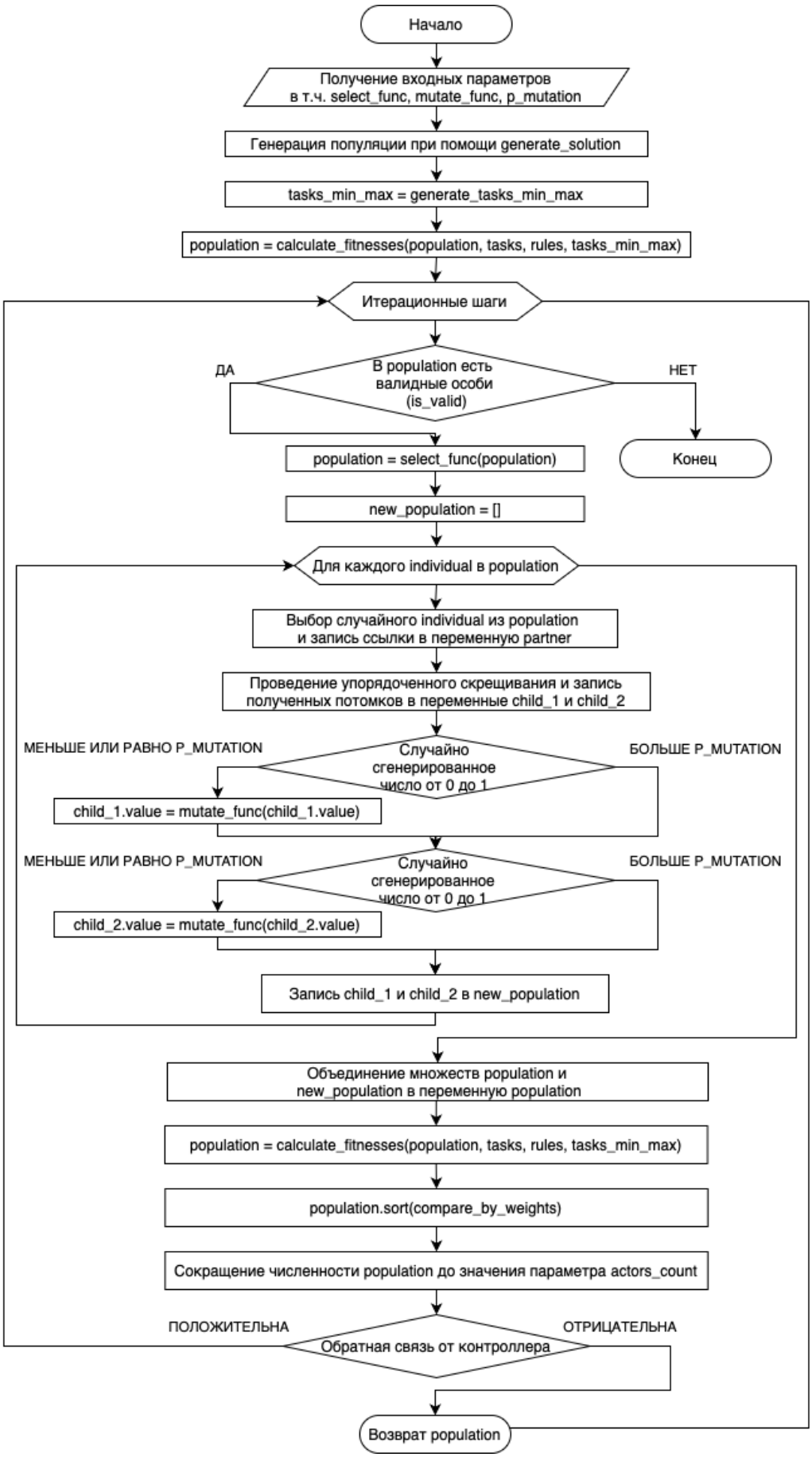


Рисунок 3.20 – Схема метода run генетического алгоритма

Последними шагами являются объединение population и new_population в переменную population, проводится пересчет итоговой приспособленности всех акторов и сокращение population до значения параметра actors_count. По завершению работы алгоритма, метод возвращает значение переменной population.

Схема алгоритма скрещивания, используемого в ГА, показана на рисунке 3.21.

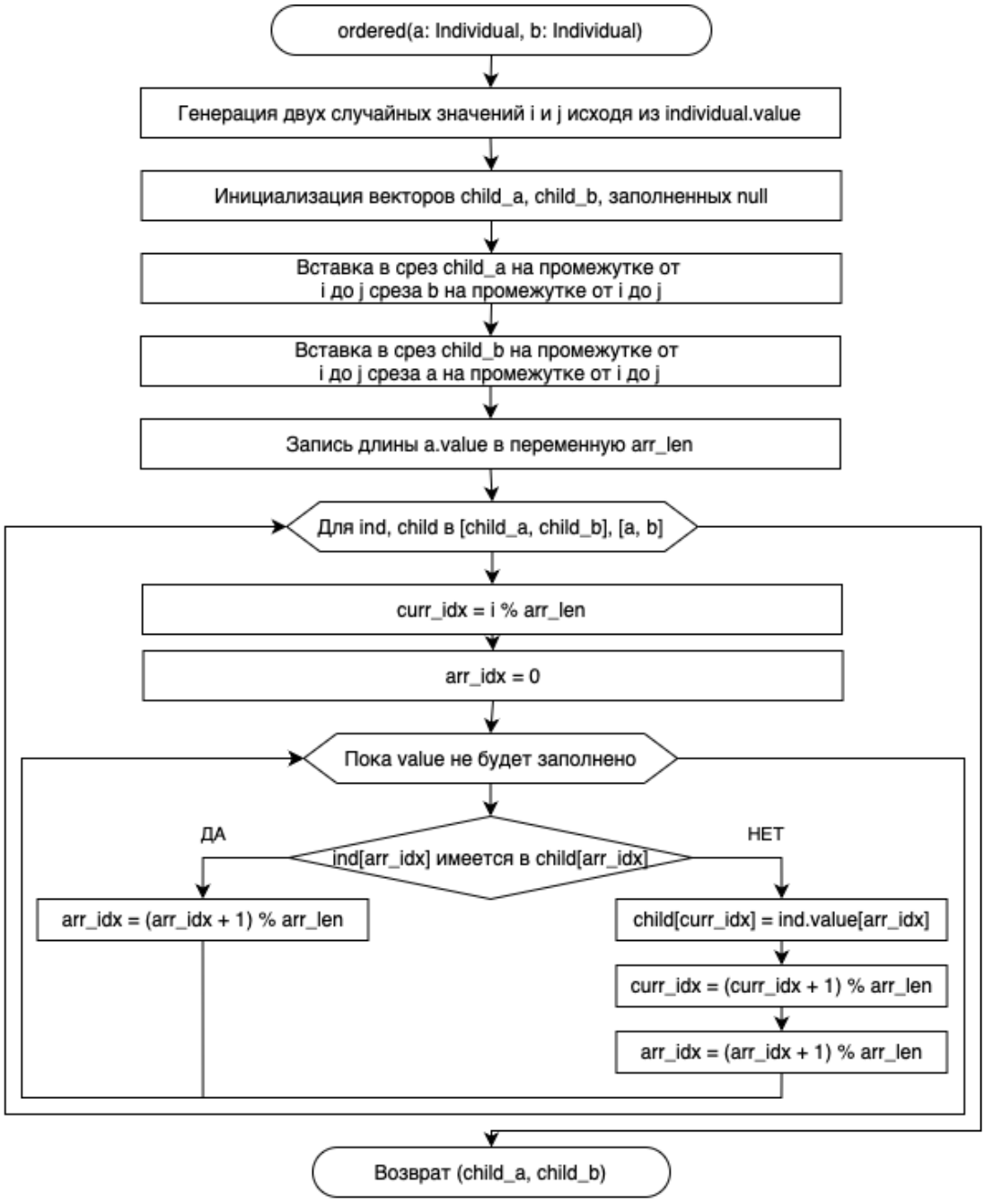
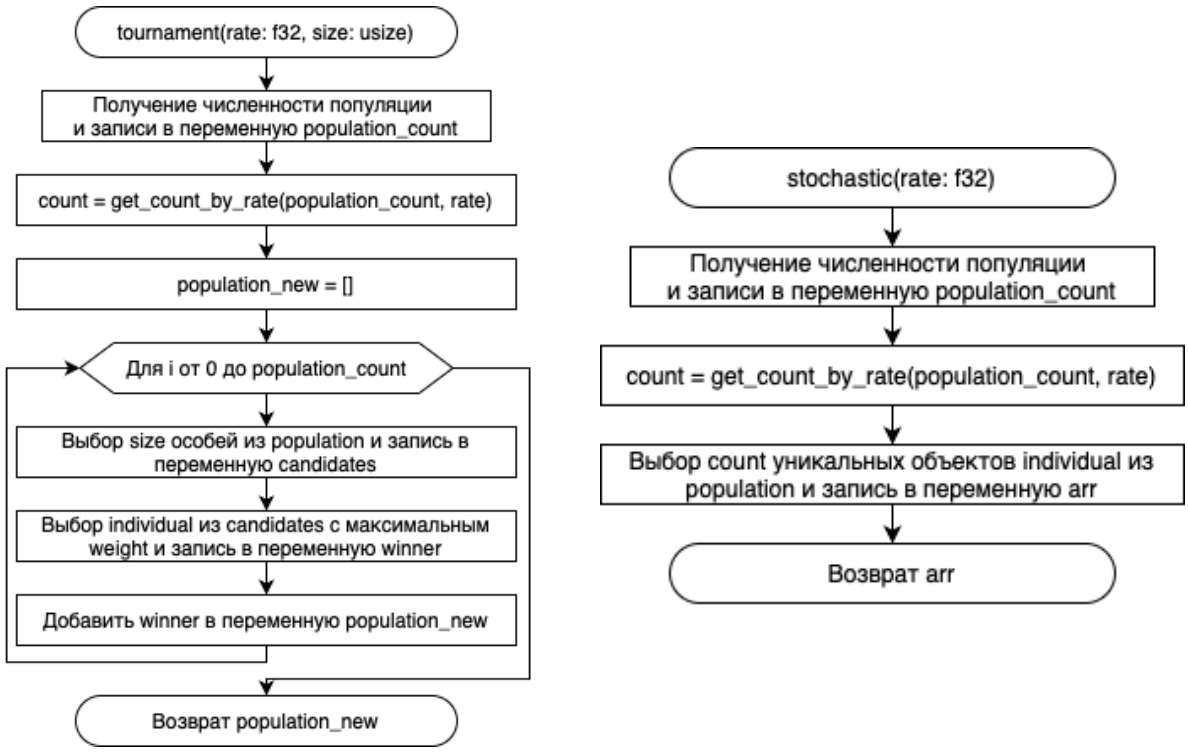


Рисунок 3.21 – Схема алгоритма скрещивания

В соответствии с данным методом скрещивания, в первую очередь происходит выбор случайного участка: случайных образом определяется два индекса (i и j), которые задают начальную и конечную позиции подмножества генома (значения *x*). Это подмножество присваивается потомкам по тем же индексам аналогично двуточечному методу скрещивания.

В child_a оставшиеся позиции заполняются элементами из родителя (переменная b) начиная с индекса j + 1, затем продолжается с нулевого индекса, исключая те, что уже присутствуют в child_a. Аналогичный процесс проводится по отношению к child_b, однако элементы множества копируются от другого родителя (под переменной a). Функция возвращает child_a и child_b.

Схемы алгоритмов отбора «Турнирный» и «Стохастический» показаны на рисунке 3.22а и рисунке 3.22б соответственно.



а) б)

Рисунок 3.22 – Схема алгоритмов отбора:

а) Турнирный;

б) Стохастический

Стоит отметить, что в программном коде эти два методы представляют собой каррированные функции: внешняя функция получает аргументы rate (доля популяции, которая должна «выжить» после отбора) и size (размер турнира, у «Турнирного» метода отбора). Возвращаемая функция получает на вход множество типа Population и возвращает множество с тем же типом, содержащее отобранных акторов.

Функция tournament начинает свою работу с вычисления численности особей, подлежащих отбору на основе размера множества и аргумента rate, путем вызова get_count_by_rate. Результат вычисления численности записывается в population_count. Также, инициализируется переменная, отвечающая за множество отобранных особей (population_new).

Далее, в теле цикла от 0 до population_count случайным образом выбирается множество особей в количестве значения size, и особь с наибольшим МКС добавляется в population_new. После цикла, функция возвращает значение переменной population_new.

Функция stochastic вычисляет численность особей, подлежащих отбору, аналогично функции tournament, инициализирует множество population_new и заполняет его уникальными особями из population случайным образом, пока численность population_new не станет равна population_count.

Схема алгоритма метода отбора «Рулетка» показана на рисунке 3.23.

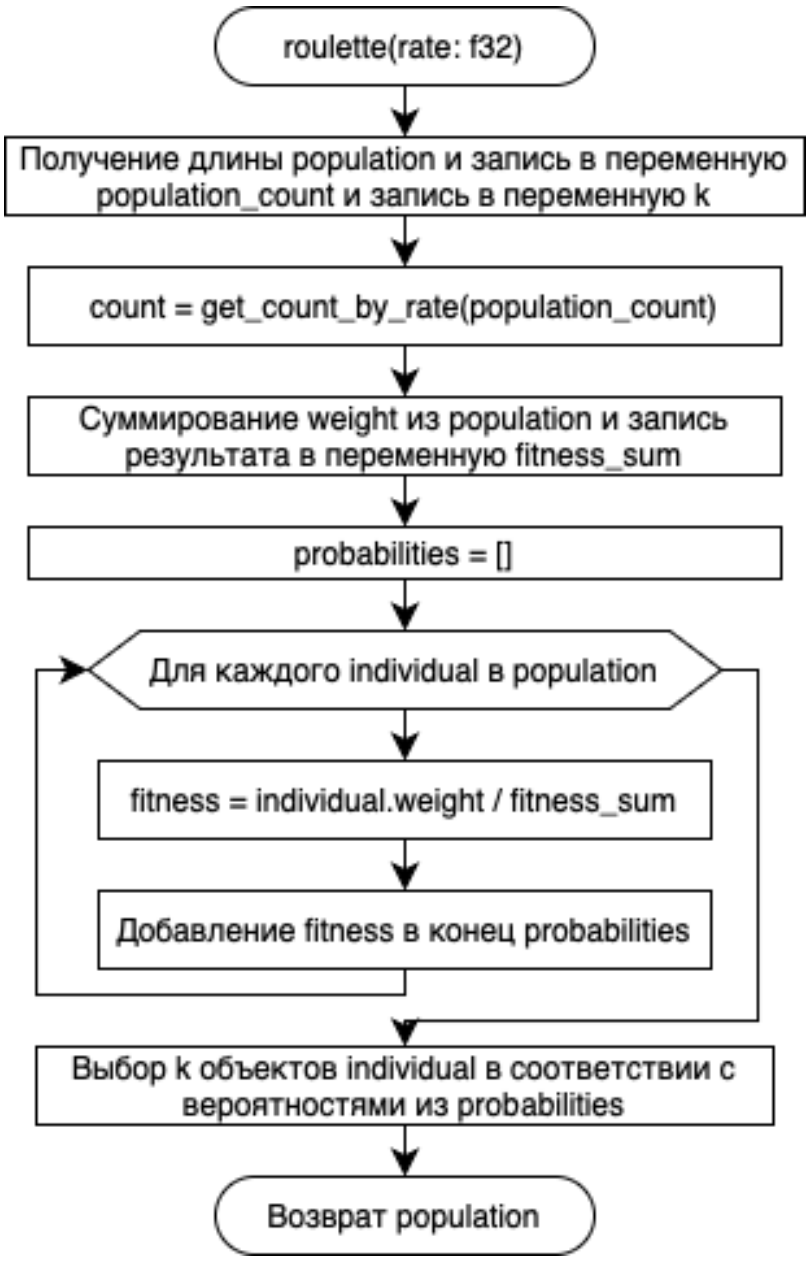


Рисунок 3.23 – Схема алгоритма метода отбора «Рулетка»

Аналогично методам tournament и stochastic метод roulette вычисляет численность особей, подлежащих отбору, и записывает результат в переменную count. Далее, значения weight всех особей суммируются, а результат вычисления записывается в переменную fitness_sum. Также инициализируется множество probabilities для вычисления вероятностей.

Для каждой особи (актора) в population вычисляется вероятность путем вычисления частного между значением weight и fitness_sum, результат записывается в probabilities. Функция возвращает множество объектов Individual в размере count с вероятностями, вычисленными в probabilities.

Схема метода run ПА показана на рисунке 3.24.

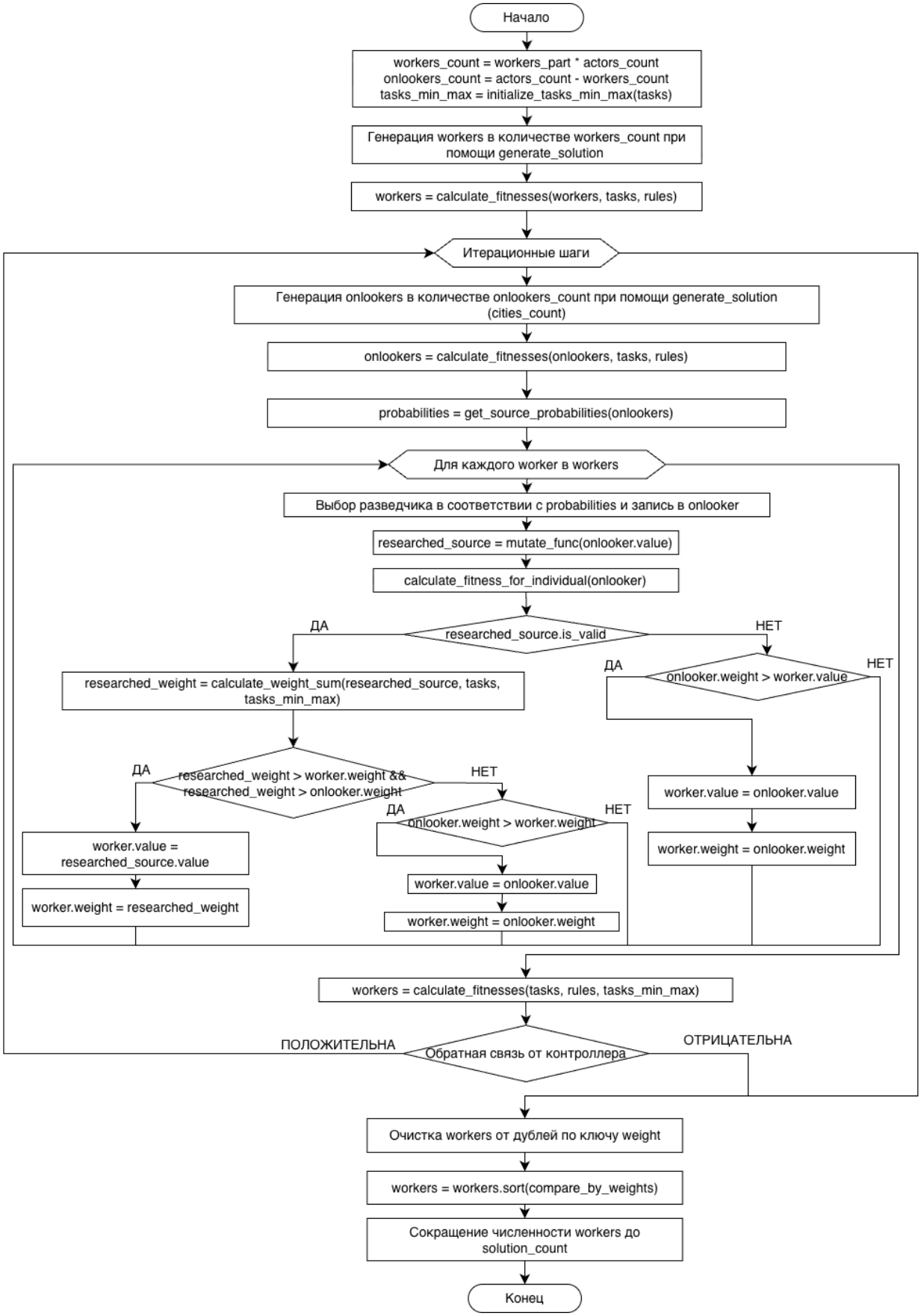


Рисунок 3.24 – Схема метода run пчелиного алгоритма

В первую очередь, на основе параметров actors_count и workers_part вычисляются численности разведчиков (onlookers) и рабочих (workers), а также инициализируется tasks_min_max. Далее, генерируется множество workers, и для ведется вычисление их приспособленности при помощи метода calculate_fitnesses.

Далее идет итерационный цикл. В теле цикла, аналогично workers, генерируется множество onlookers и проводится вычисление их приспособленности. После этого, приспособленность повторно вычисляется для workers. Это необходимо для актуализации значений МКС т.к. один из разведчиков мог обновить tasks_min_max, тем самым сделав МКС рабочих неактуальными, что привело бы к некорректному сравнению в дальнейшем. На основе МКС значений разведчиков, формируется вероятностное множество probabilities, которое отвечает за вероятность рабочих «присоединиться» к тому или иному разведчику. Множество probabilities формируется при помощи метода get_source_probabilities. Блок-схема алгоритма показана на рисунке 3.25.

Далее, в теле цикла идет прогон по каждому рабочему из множества workers. Здесь рабочий «присоединяется» к одному из разведчиков, а затем «мутирует» копию его решения в соответствии с выбранной функцией мутации. Соответственно, для мутированного решения вычисляется МКС при помощи calculate_fitness_for_individual. Далее, рабочий выбирает для себя решение, основываясь на МКС разведчика, его мутированного варианта и собственного. Если мутированное решение не удовлетворяет ограничениям (невалидное), оно не участвует в этом сравнении. После «прохода» по рабочим, для множества workers заново вычисляется приспособленность при помощи calculate_fitnesses. На также заканчивается тело итерационного цикла.

По окончанию итерационного цикла или его прерыванию модулем контроля эффективности решения, множество workers, сортируется по МКС, удаляются дубликаты, сокращается до значения solutions_count и возвращается из метода run.

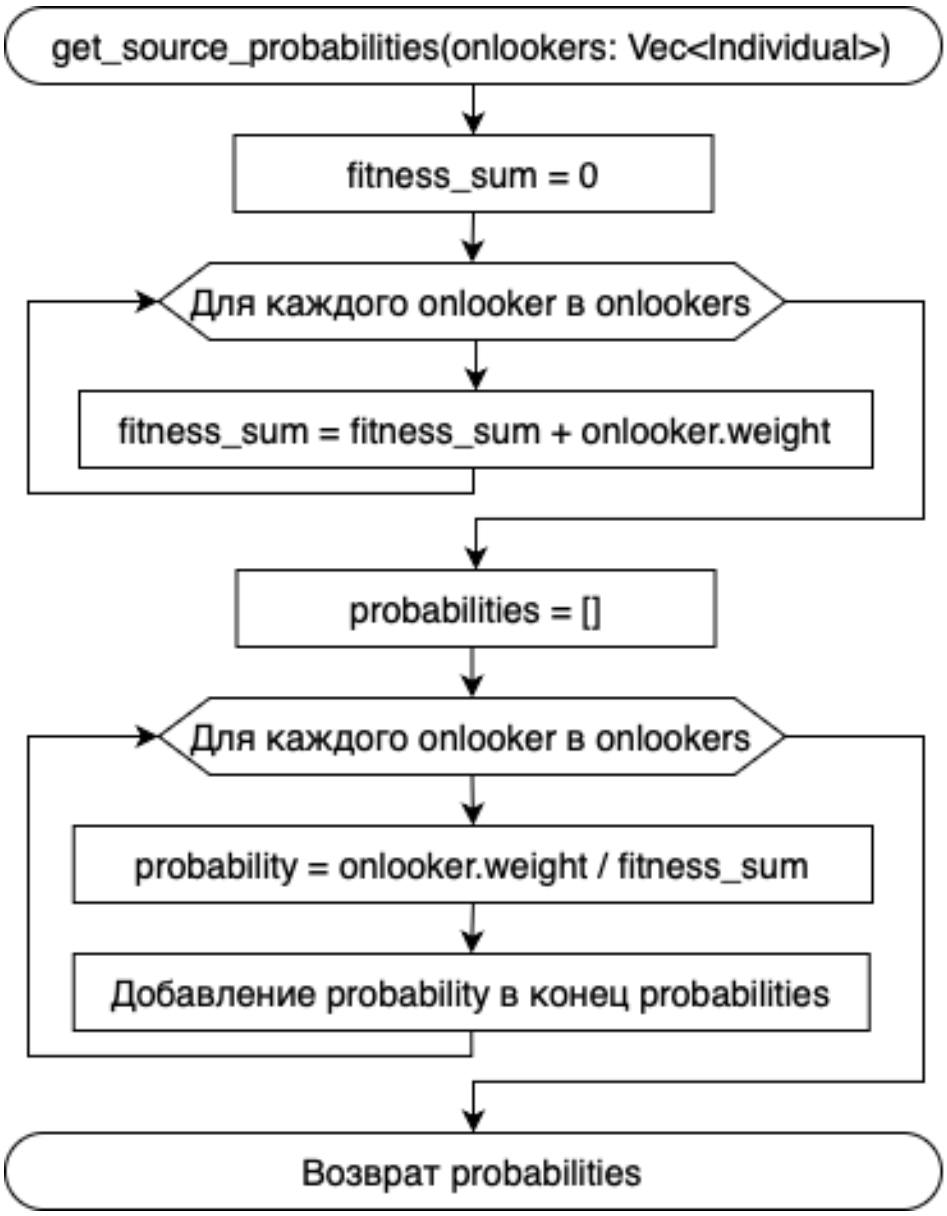


Рисунок 3.25 – Схема алгоритма get_source_probabilities

Схема метода run ИО показана на рисунке 3.26.

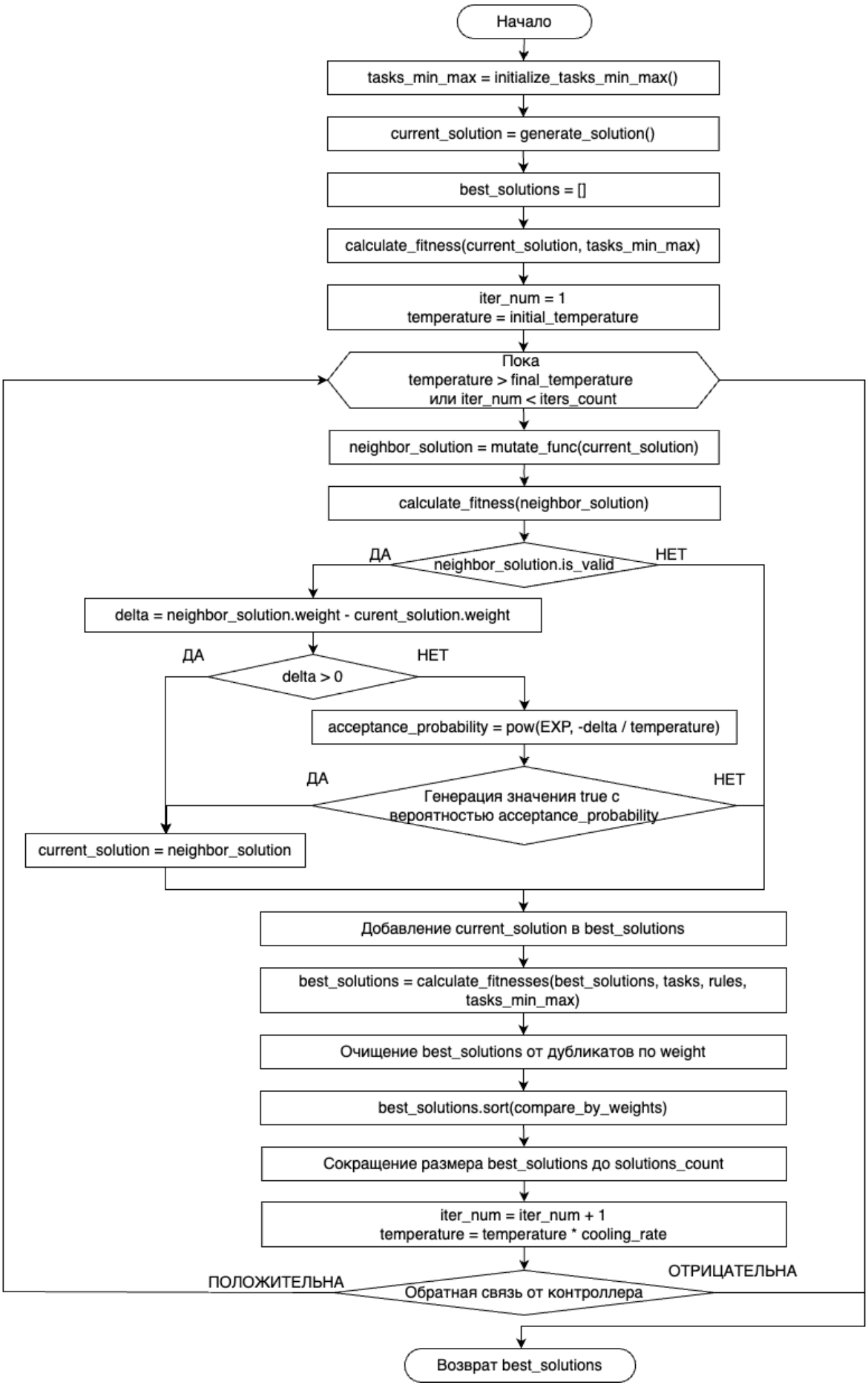


Рисунок 3.26 – Схема метода run алгоритма имитации отжига

Помимо общих параметров, ИО принимает на вход метод мутации, начальную температуру, конечную температуру и коэффициент охлаждения. Параметр iters_count является опциональным, то также может выступать критерием для остановки алгоритма.

В программной реализации ИО в первую очередь генерируются tasks_min_max, инициализируются начальное состояние температуры (temperature) и счетчик итерационных шагов (iter_num). Затем инициализируется начальное решение, которое записывается в переменную current_solution. Также, для него вычисляется МКС с использованием функции calculate_fitness.

Далее, идет итерационный цикл, который идет до тех пор, пока temperature не станет меньше final_temperature или iter_num не станет меньше iters_count (если задано). В теле цикла создается новое решение (neighbor_solution), которое является результатом работы mutate_func с аргументом current_solution. Для нового решения вычисляется МКС и расчитывается разница delta. Если новое решение лучше (delta положительно), оно принимается сразу. В обратном случае, оно будет принято с некоторой долей вероятности. Если новое решение принято, оно присваивается переменной current_solution, а current_solution добавляется в множество лучших решения best_solutions. В конце тела цикла, в best_solutions пересчитываются значения приспособленности, множество обрезается до значения параметра solutions_count. Также, обновляются счетчики: temperature умножается на cooling_rate а iter_num инкрементируется. Здесь же модуль контроля эффективности решений может прервать цикл.

Когда итерационный цикл завершается, best_solutions очищается от дубликатов по МКС и сортируется по этому же значению.

3.6 Реализация модуля интеллектуального построения стратегий

3.6.1 Анализ данных

Для обучения интеллектуальной модели, решено сгенерировать постановки задачи ЦУО и прогнать их через конфигурации используемых эволюционных алгоритмов. В целях определения перечня используемых конфигураций, было сделано множество вычислительных экспериментов с разными размерностями задачи, численностью критериев, перечнем постановок и т.д.

В первую очередь проведен эксперимент задачами без ограничений [82]. Постановки задач коммивояжера получены из берлинского университета Цузе [83], но разбиты на задачи размерностью 15-20 точек. В результате разбиения, для эксперимента было получено 10 тыс. постановок задачи. Поскольку добавление в исследование ИО было проведено позднее данного эксперимента, конфигурации данного алгоритма не представлены в результатах эксперимента.

На рисунке 3.27 показан график, отображающий, численность задач, в которых та или иная конфигурация показала наибольшее значение МКС. В целях предотвращения «перегрузки» графика, были оставлены первые 20 модификаций. В рамках данного эксперимента использовались другие сокращения эволюционных алгоритмов: МА – МУР, ПА – ПЧЕ, ГА – ГЕН).

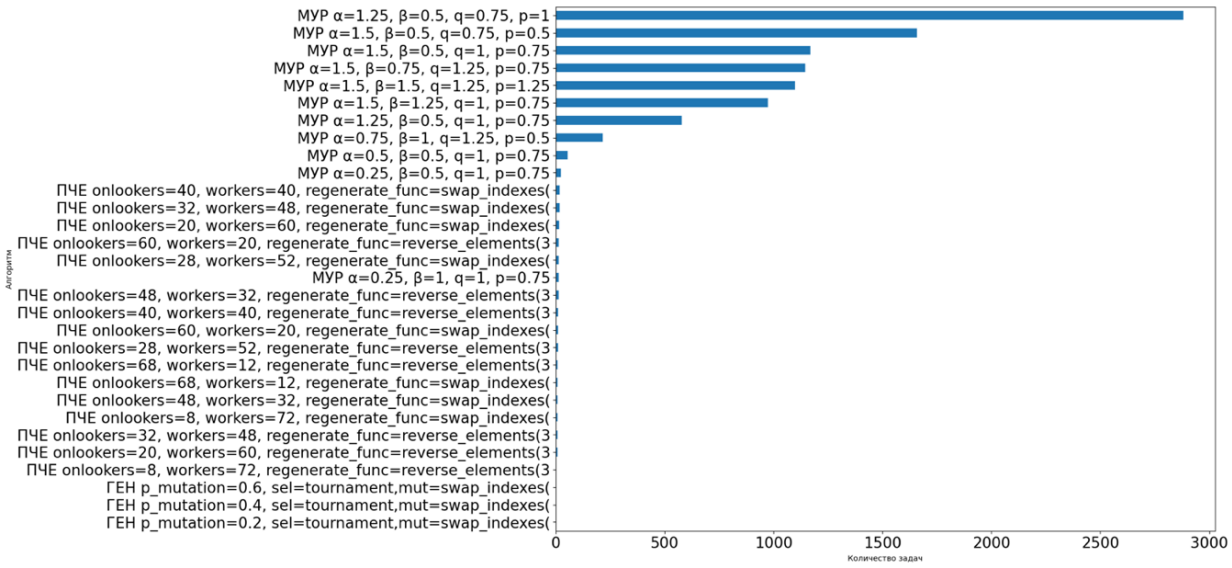


Рисунок 3.27 – Диаграмма эффективности конфигураций эволюционных алгоритмов по количеству решенных задач

В большинстве задач наибольший показатель МКС показали конфигурации МА, далее следуют ПА, тогда как ГА оказался эффективным лишь на ограниченном числе постановок.

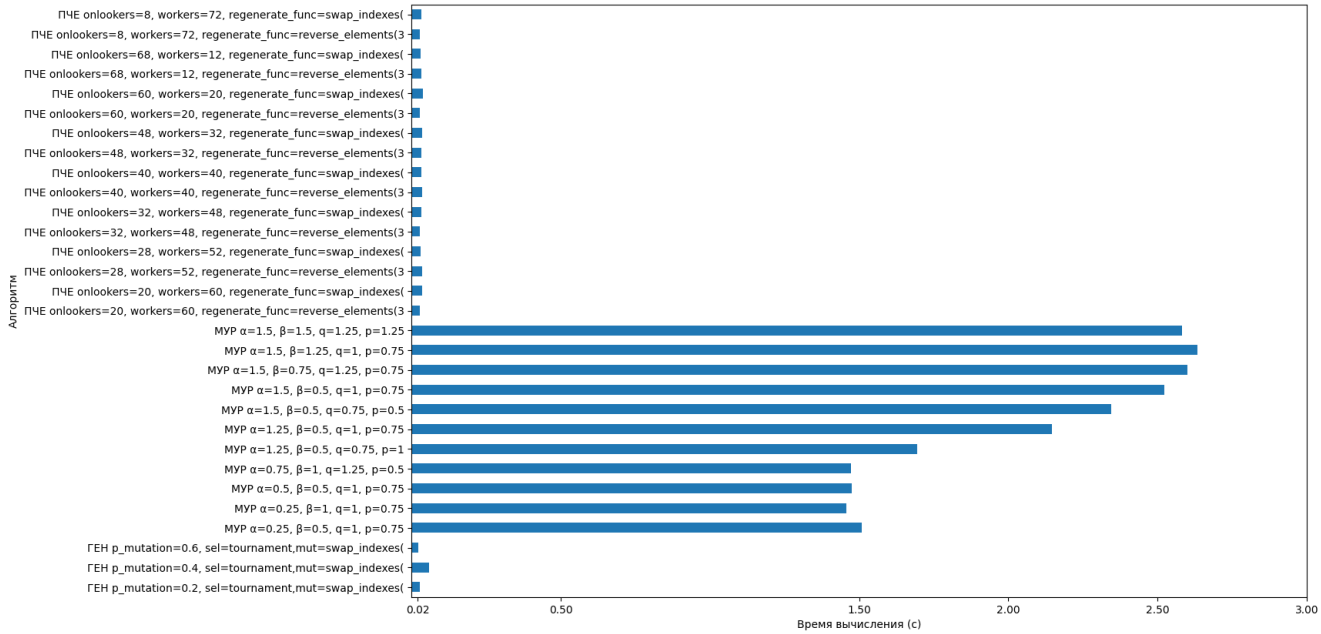


Рисунок 3.28 – Диаграмма медианного значения вычислительного времени конфигураций эволюционных алгоритмов

Согласно медианному времени вычислений (рисунок 3.28), МА демонстрирует существенно более длительное время работы — вплоть до двух порядков медленнее ГА и ПА. На рисунке 3.29 показано, что без учета МА некоторое превосходство по скорости имеют отдельные конфигурации ГА по сравнению с ПА.

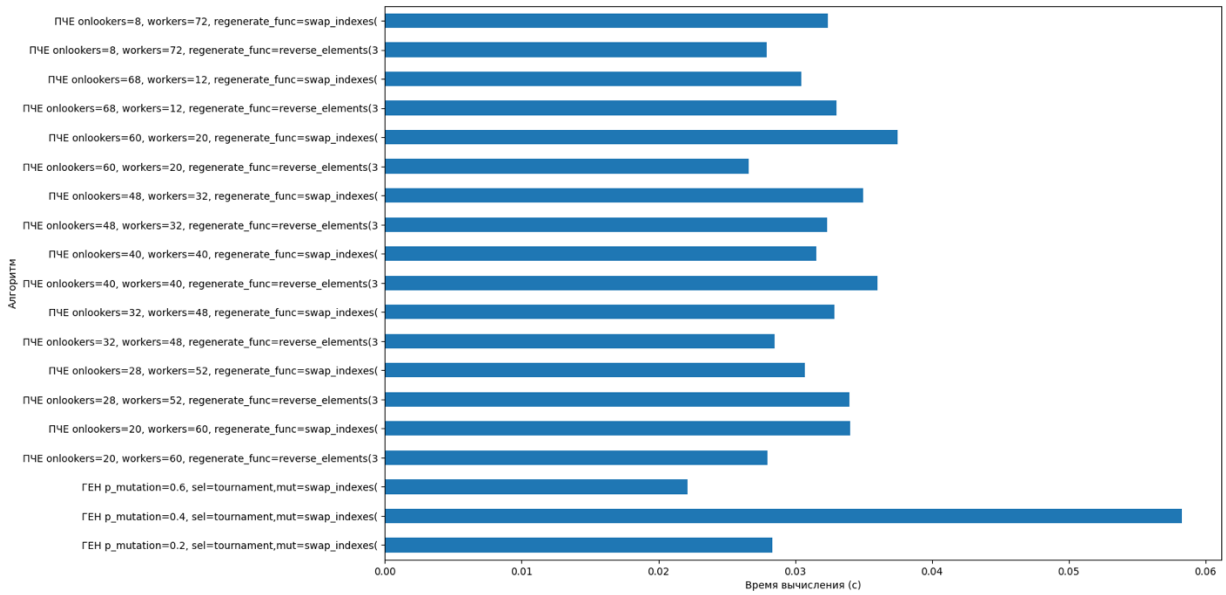


Рисунок 3.29 – Диаграмма медианного значения вычислительного времени конфигураций эволюционных алгоритмов за исключением муравьиного

На рисунке 3.30 отражено количество задач, где каждая конфигурация оказалась наиболее быстрой. Лидерами стали ГА и ПА, что согласуется с их низкой вычислительной сложностью. Далее анализировалась скорость сходимости (рисунок 3.31): наиболее быстро достигают локального максимума МА и ГА.

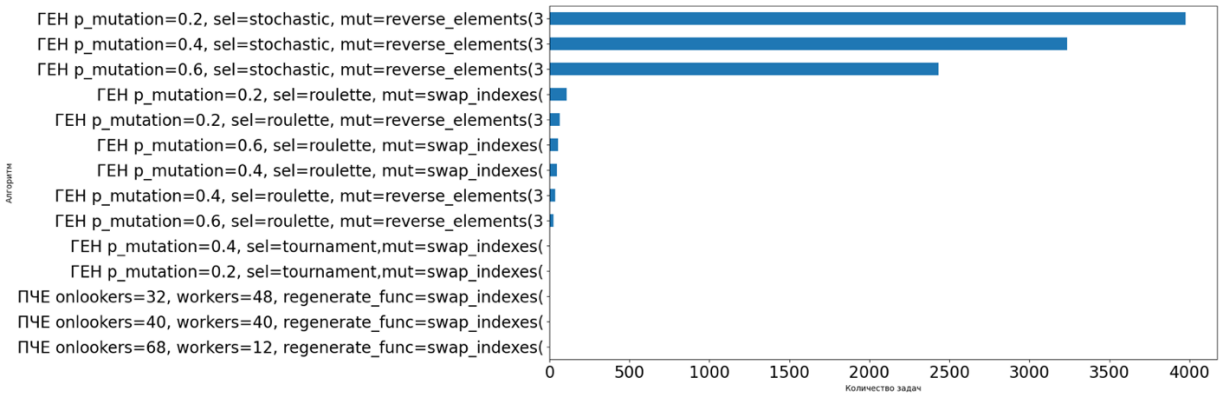


Рисунок 3.30 – Диаграмма эффективности алгоритмов по скорости решения задач

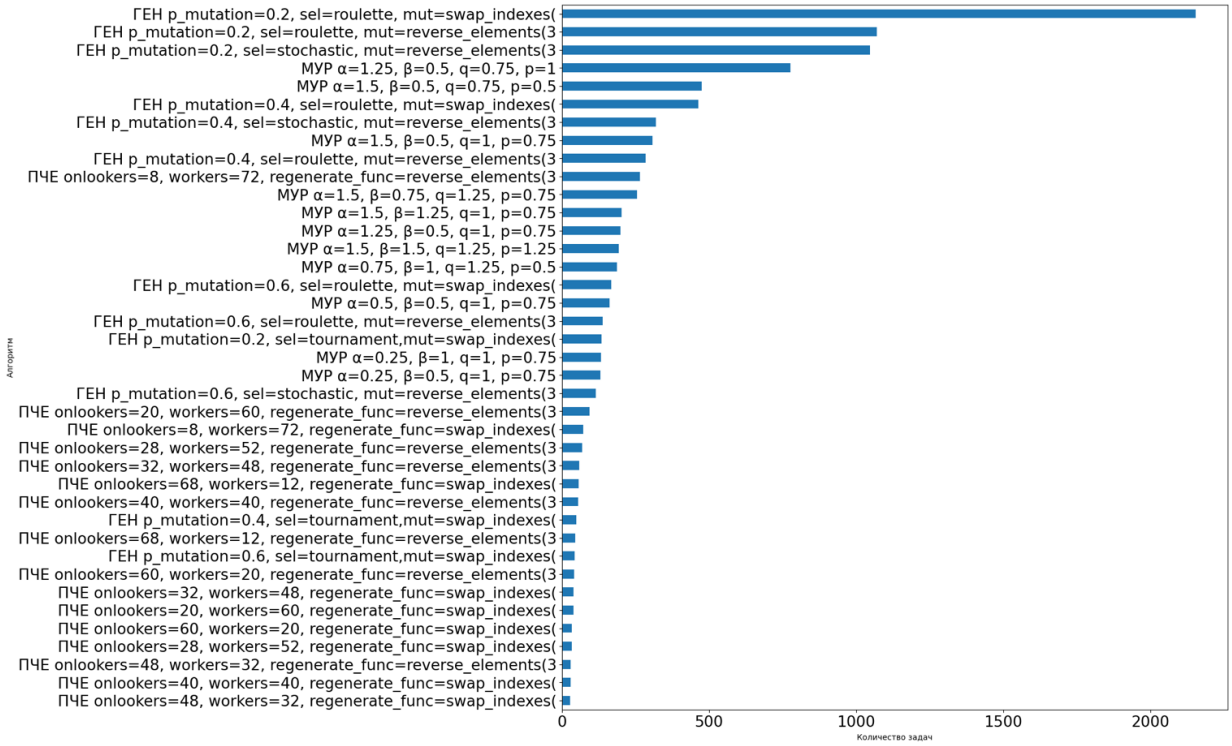


Рисунок 3.31 – Диаграмма эффективности алгоритмов по скорости сходимости

Средние и медианные скорости (рисунок 3.32 и рисунок 3.33) подтверждают преимущество ГА в ранней сходимости и устойчивость показателей МА.

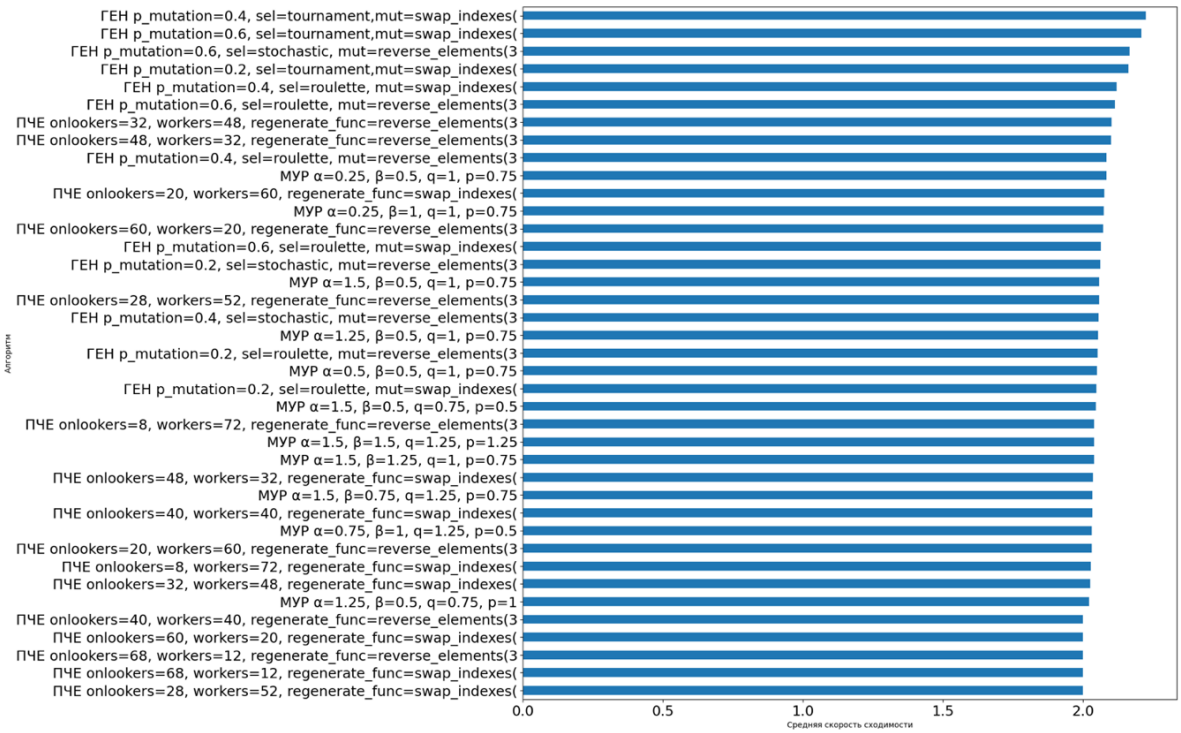


Рисунок 3.32 – Диаграмма средних скоростей сходимости алгоритмов

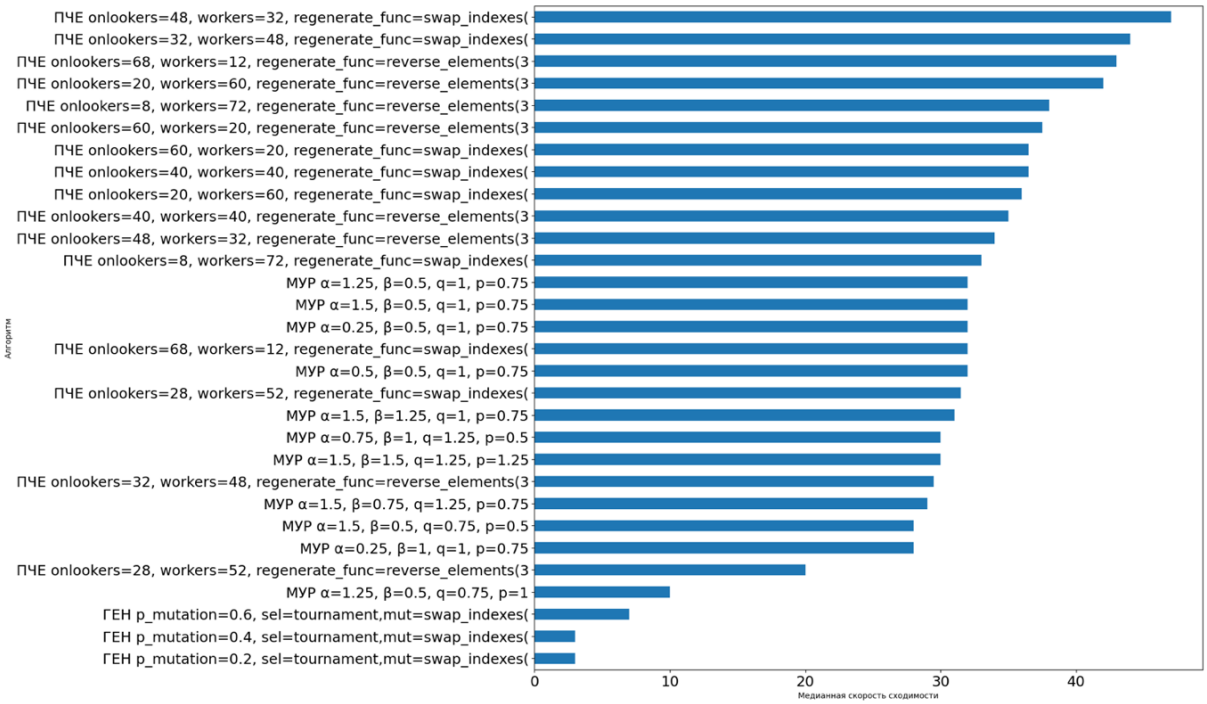


Рисунок 3.33 – Медианная скорость сходимости лучших в нахождении минимума алгоритмов

В следующем эксперименте [84] в каждой задаче использовалось 2 критерия: расстояние и время, а также заданы ограничения, эмулирующие доступные временные окна. Ограничения представлены на таблице 3.3.

Таблица 3.3 – используемые в эксперименте правила ограничений

№	Текст ограничения	Штраф
1	*# A ~ [1,5] & # D ~ [8,15] & @ A D время > 3000*	время: 5000
2	*# B = [1,3] & @ B F время < 1200*	исключение
3	*@ C H время < 200 | @ C H время > 2600*	исключение
4	*# E = [8,15] & @ E K время > 3500*	время:9000
5	*# L = [1,4] & # M = [5,10] & (@ L M время < 150 | @ L M время > 220)*	исключение

Из-за установленных ограничений (временных окон), не все постановки задач представилось возможным решить: из 30 тыс. постановок удалось решить около 23 тыс. На основе них будет представлен дальнейший анализ полученных результатов. На таблице 3.4 приведены максимальные и минимальные значения критериев оптимизации.

Таблица 3.4 – значения критериев оптимизации, полученные в результате вычислений

Критерий	Минимальное значение	Максимальное значение
Расстояние	223,0	1304101,49
Время	184,4	976847,12
Средневзвешенное	0,0000041	0,9914713

На рис. 3.34 показано распределение лидирующих конфигураций: наиболее эффективными оказались МА с $\alpha > 1$ и умеренной жадностью $(0{,}5 \leq \beta \leq \alpha)$. На втором месте — ГА с методом отбора «Турнирный» и мутацией РП. Конфигурации ПА заняли третье место, при этом взаимосвязи параметров выявить практически не удалось.

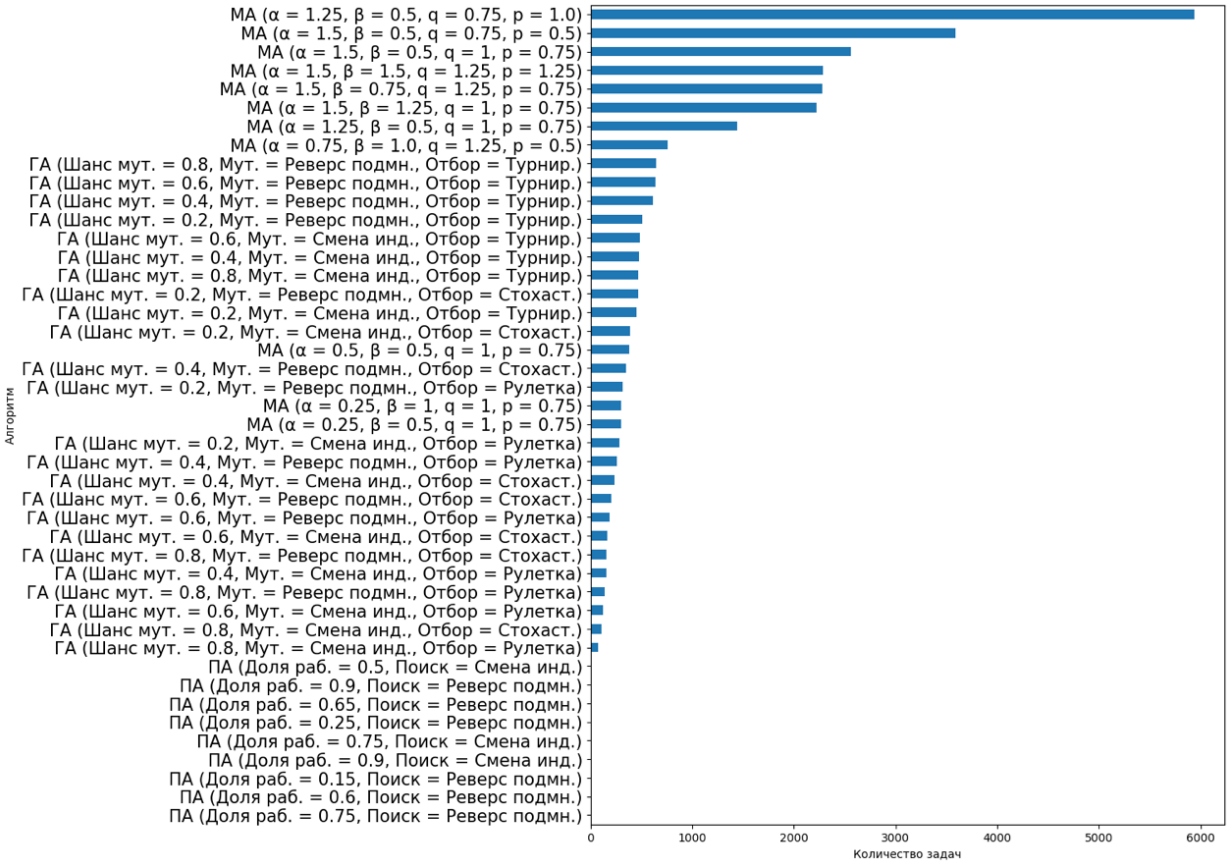


Рисунок 3.34 – График численности задач, в которых та или иная конфигурация эволюционного алгоритма получила наибольший показатель межкритериальной суммы

Рисунок 3.35 демонстрирует медианное время решения: наиболее быстрыми остаются ГА, причём конфигурации с Рулеткой и Стохастическим отбором работают примерно вдвое быстрее турнирных. МА существенно медленнее всех, а ПА демонстрирует высокую вариативность времени, что объясняется малым числом задач, в которых он оказался лучшим.

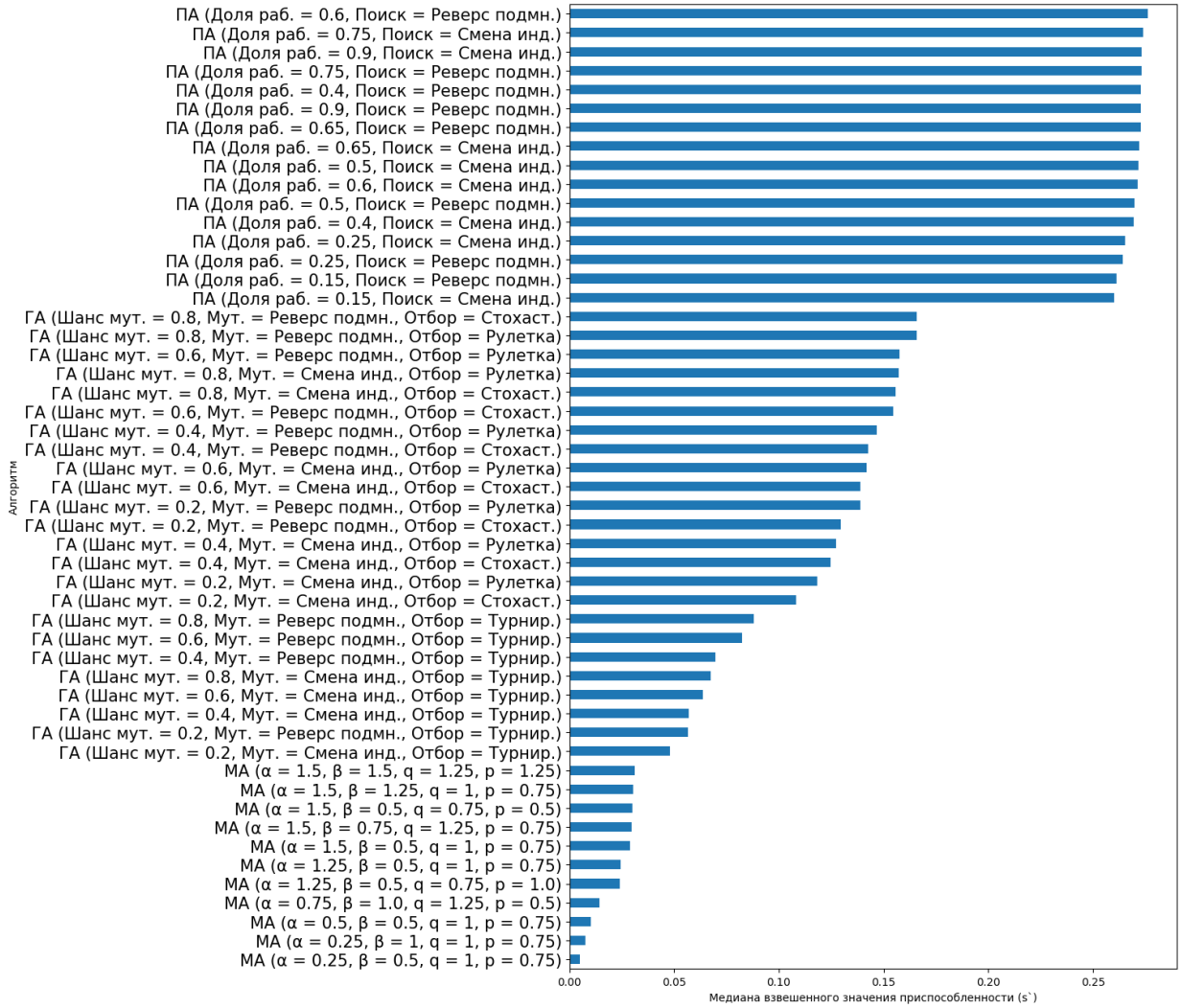


Рисунок 3.35 – график эффективности конфигураций эволюционных алгоритмов оптимизации по медианному значению межкритериальной суммы

Проведенный вычислительный эксперимент позволил выявить наиболее эффективные конфигурации ЭА для решения задач ЦУО в логистике. В ходе исследования установлено, что конфигурации МА продемонстрировали наибольшую эффективность по качеству решений. Высокая эффективность конфигураций МА связана с оптимальным выбором параметров, при которых коэффициенты феромона играют значимую роль в решении задачи.

ГА занял второе место по эффективности, проявив наибольшую производительность в условиях оптимизации времени выполнения. Этот класс

алгоритмов особенно выгоден в задачах, где скорость решения критична, что подтверждается результатами медианного времени вычислений.

Конфигурации ПА имеют менее выраженные результаты, но в ряде случаев также продемонстрировали высокую эффективность, что делает их конкурентоспособными в определенных постановках задач.

Таким образом, результаты эксперимента подтверждают целесообразность использования конфигураций МА и ГА для решения задач оптимизации в логистике. Более того, эти алгоритмы могут быть использованы для дальнейшего обучения моделей нейронных сетей, что откроет дополнительные возможности для автоматизации и повышения эффективности логистических процессов.

Следующий эксперимент включал в себя большее количество постановок задачи, также использование конфигураций ИО [85]. В первую очередь было проанализировано соотношение вычислительного времени и МКС. Третий эксперимент включал алгоритм ИО. На графиках (рисунок 3.36) видно, что рост времени выполнения редко приводит к улучшению качества, особенно для ГА и ИО. Это подтверждает необходимость механизма переключения алгоритмов при отсутствии прогресса.

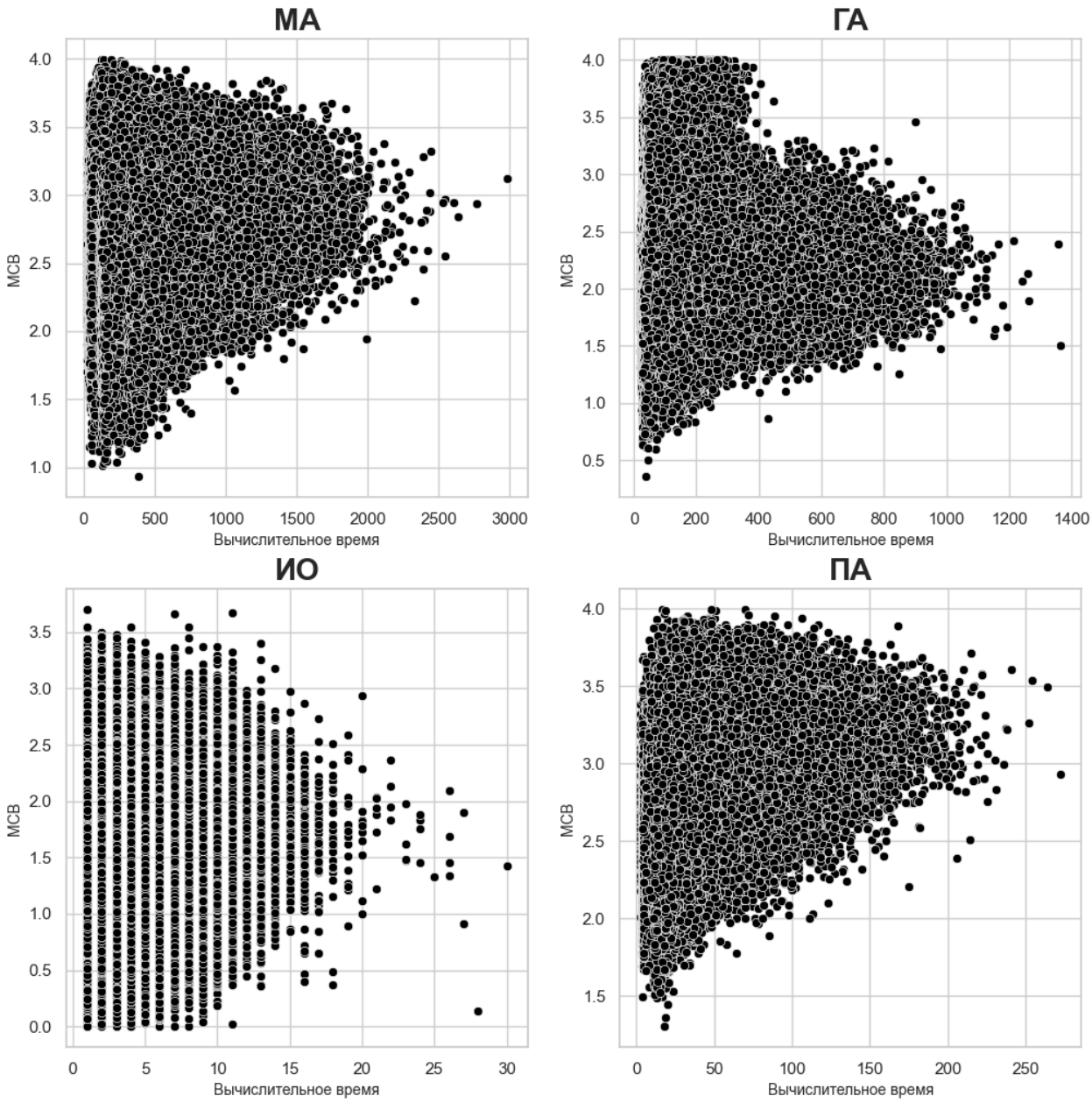


Рисунок 3.36 – Графики разброса вычислительного времени и МКС по алгоритмам

На рисунке 3.37 приведены размахи МКС. На данном графике вертикальная линия по центру прямоугольника («ящика») указывает на медианное значение, левый и правый края – первый и третий квартиль, а левый и правый край линии – минимальные и максимальные значения. Отдельными точками на графиках представлены аномальные значения (выбросы) [86]. Медианные значения большинства конфигураций лежат в диапазоне 2.0–3.0, а большое число выбросов отражает нестабильность ряда алгоритмов. График

вычислительного времени (рисунок 3.38) показывает, что МА работает дольше других, но отличается высокой стабильностью. Для ГА и ПА характерна большая вариативность как качества, так и времени.

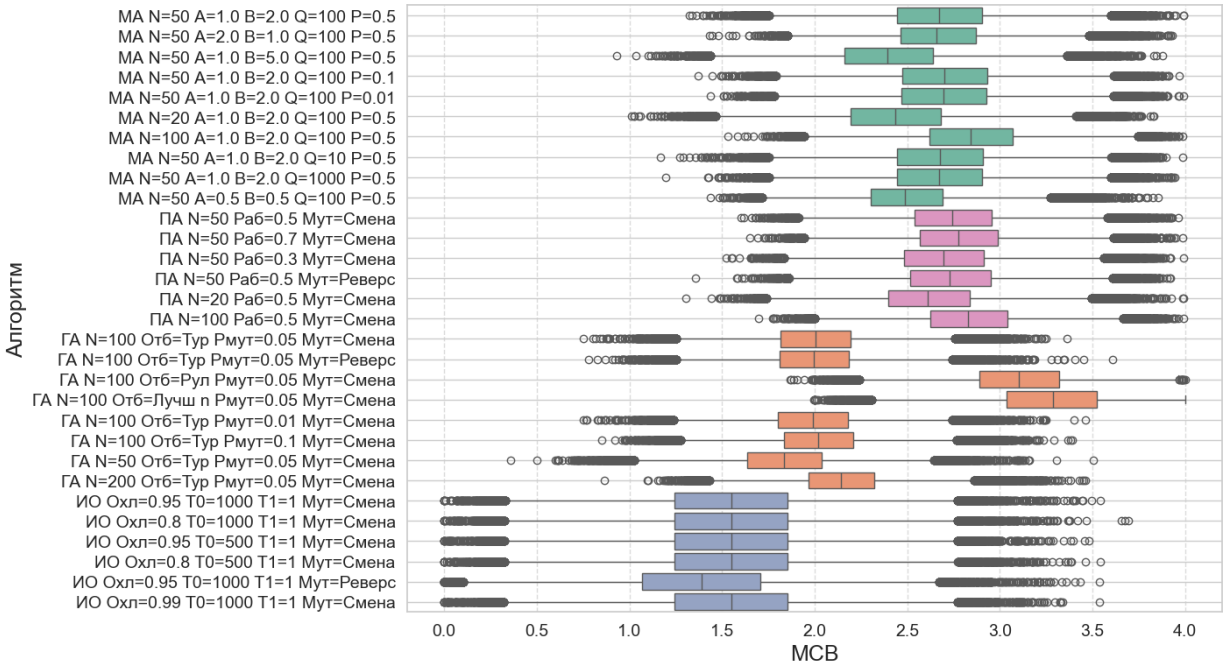


Рисунок 3.37 – График размаха МКС по алгоритмам

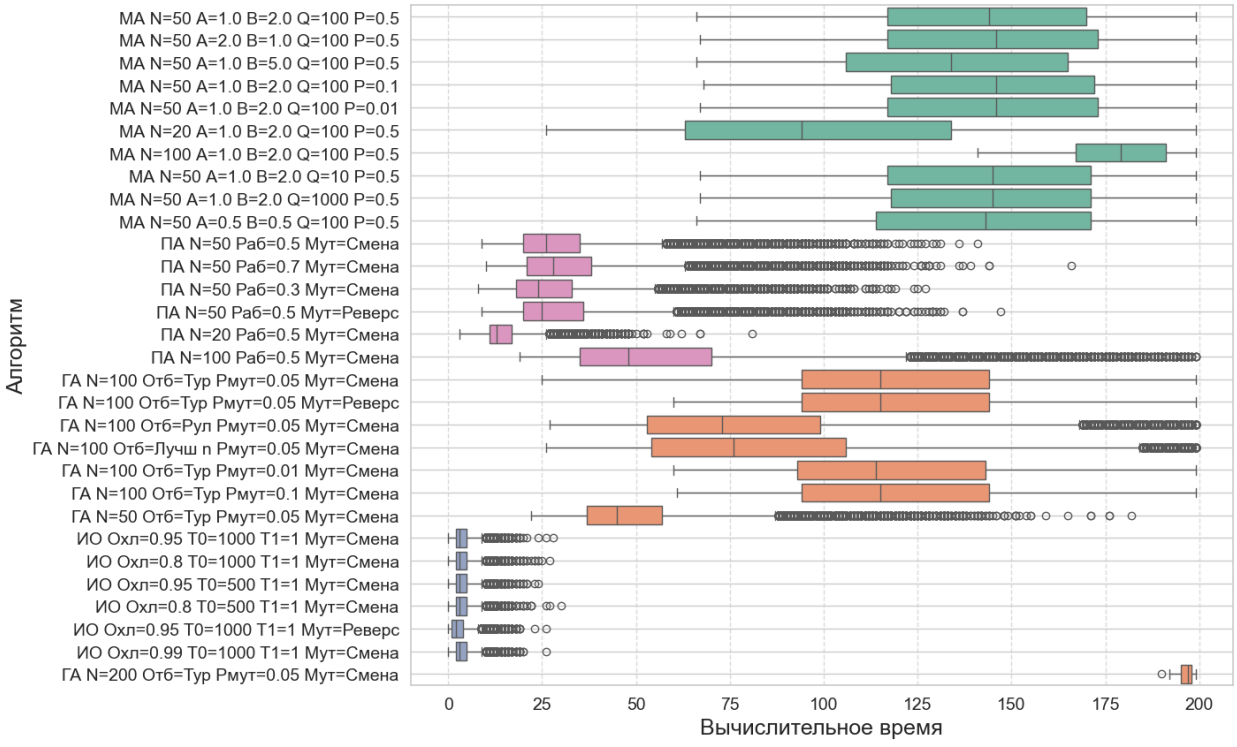


Рисунок 3.38 – График размаха вычислительного времени по алгоритмам

Анализ параметров выявил, что для МА оптимальны $\alpha \in [1; 2]$ и $\beta = 1$ и численность акторов 50–100, для ПА — метод мутации «Смена индексов» и доля рабочих пчёл >0,5, для ГА — методы отбора «Рулетка» и «Лучшие N», шанс мутации около 0.05 и мутация СИ, а ИО показал низкую вариативность и требует усложнения конфигураций.

В одном из последующих экспериментов [87] проведен анализ работы ЭА на задачах ЦУО высокой размерности (1500 – 2000 точек). Поскольку в открытых источниках отсутствуют достаточные по объему и сложности тестовые данные для задач численностью в 2 тыс. точек, было принято решение о генерации собственных наборов на основе случайных матриц смежности. Такой подход позволяет контролировать диапазоны весов, плотность связей и симметричность матриц и, соответственно, создать условия близкие к реальным сценариям (логистические сети, планирование, оптимизация конечных автоматов и др.)

Схема алгоритма генерации матриц смежности показана на рисунке 3.39. В данном алгоритме широко используется генератор случайных чисел. В первую очередь, генерируется значение, отвечающее за размерность генерируемой задачи – целое число в диапазоне [5000; 20000]. Далее, определяются диапазон значений матрицы смежности – чисел, отражающих числовое отношение между точками. Правая граница имеет диапазон [100; 1.000.000], левая граница – от 50 до значения правой границы.

Так как рассматриваемые задачи дискретной оптимизации могут рассматриваться в различных областях, производилась генерация как симметричных (по главной диагонали) матриц, так и несимметричных. Постановка задач с симметричными матрицами особенно актуальна для оптимизации логистики, тогда как «несимметричная» постановка может быть актуальна для других подобных задач (оптимизация конечного автомата, планирование расписания, теория игр и др.).

С вероятностью 0,5 алгоритм сгенерирует симметричную матрицу: для двух значений матрицы, симметричных по главной диагонали будет

сгенерировано одно число от левой границы до правой границы. Для несимметричной матрицы, для каждой ячейки будет сгенерировано отдельное число в пределах от левой до правой границ. Представление задачи в виде несимметричной матрицы смежности актуально, например, для оптимизации конечных автоматов.

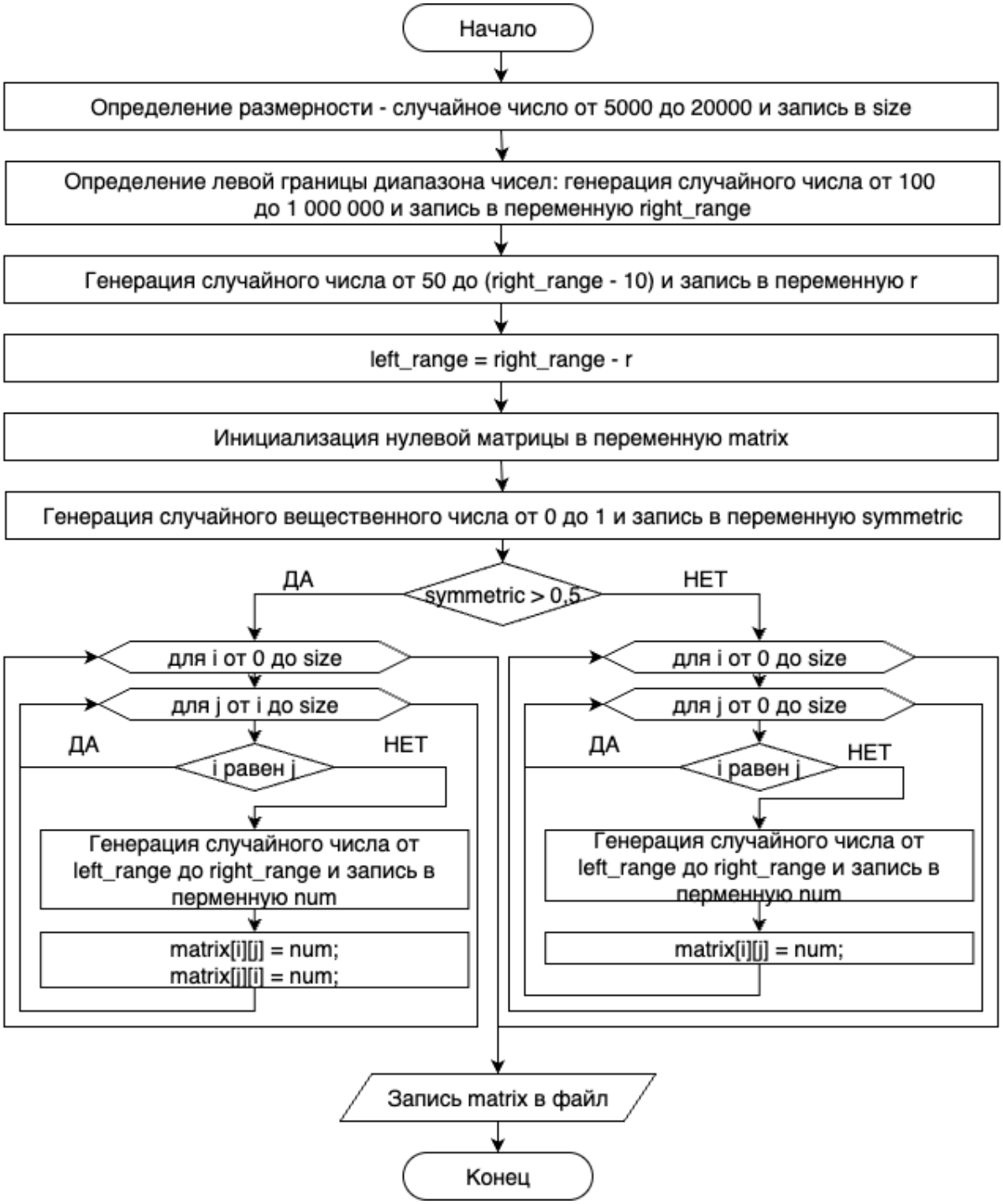


Рисунок 3.39 – Схема алгоритма генерации матрицы смежности

С вероятностью 0,5 алгоритм сгенерирует симметричную матрицу: для двух значений матрицы, симметричных по главной диагонали будет сгенерировано одно число от левой границы до правой границы. Для несимметричной матрицы, для каждой ячейки будет сгенерировано отдельное число в пределах от левой до правой границ. Представление задачи в виде несимметричной матрицы смежности актуально, например, для оптимизации конечных автоматов.

Результатом генерации матрицы является ее запись в файл для дальнейшего использования в вычислительных экспериментах. В результате сгенерировано 13313 постановок задач дискретной оптимизации, 7456 из которых представляют собой симметричные матрицы, 5857 – несимметричные. Численности постановок по размерностям (в диапазонах) проведены на таблице 3.5.

Таблица 3.5 – Численности постановок по размерностям

Диапазон	Кол-во	Диапазон	Кол-во	Диапазон	Кол-во	Диапазон	Кол-во
500-600	770	901-1000	867	1301-1400	1018	1701-1800	817
601-700	779	1001-1100	1022	1401-1500	961	1801-1900	881
701-800	864	1100-1200	957	1501-1600	909	1901-2000	888
801-900	833	1201-1300	768	1600-1700	979		

Остановка работы конфигурации алгоритма производится при выполнении одного из перечисленных ниже условий:

– отсутствие существенного (0,05 и выше) прироста МКС на протяжении 50 итерационных шагов подряд;

– достижение предельной численности итерационных шагов в 1000;

– достижение предельного вычислительного времени в 30 минут (1,8 млн мс).

На рисунке 3.40 приведен график оценки вариативности МКС по конфигурациям алгоритмов. По вертикальной оси указаны конкретные конфигурации эволюционных методов, а по горизонтальной – значения МКС.

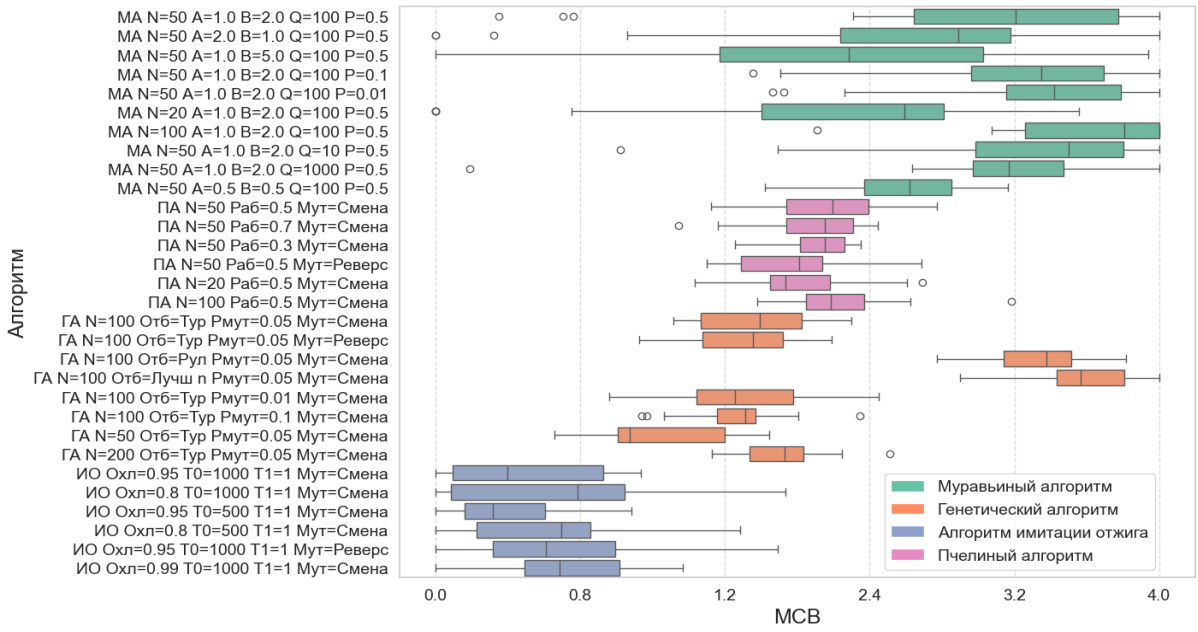


Рисунок 3.40 – Оценка вариативности МКС по конфигурациям алгоритмов

На рисунке 3.41 приведен график численностей задач, в которых выбранная конфигурация показала наибольший МКС среди других.

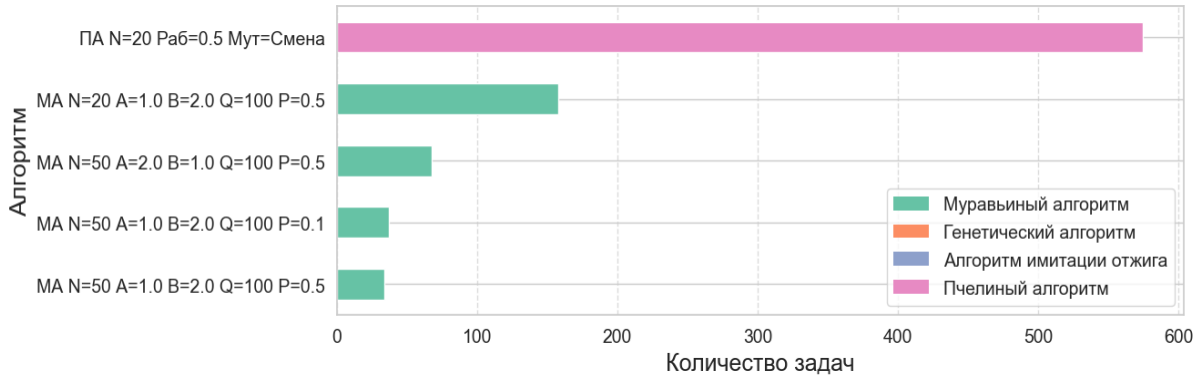


Рисунок 3.41 – Количество задач, в которых конфигурация алгоритма достигла наибольшего МКС среди других

На рисунке 3.42 приведены графики разброса вычислительного времени и МКС по алгоритмам.

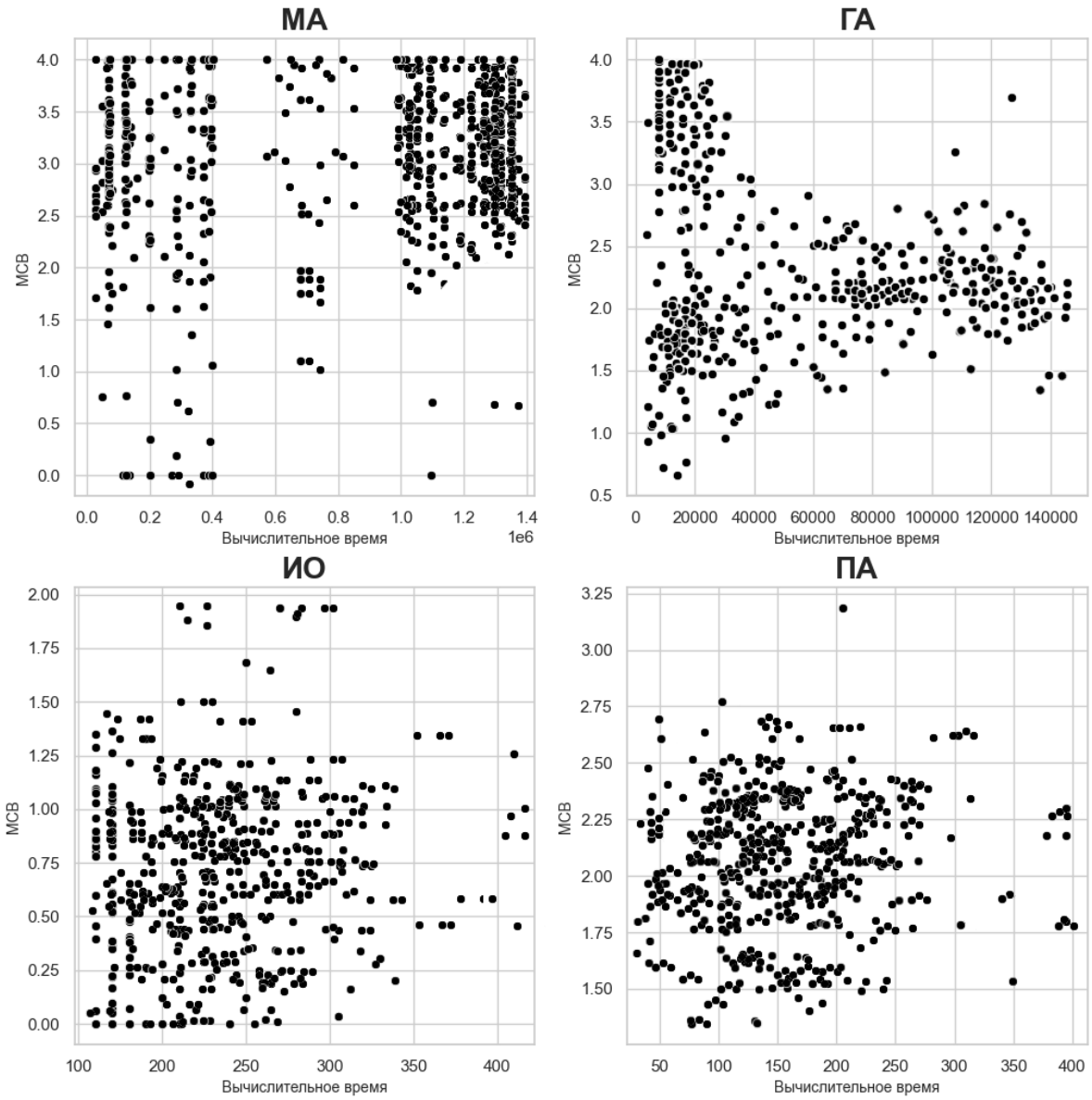


Рисунок 3.42 – Графики разброса вычислительного времени и МКС по алгоритмам

Наиболее высокие значения МКС достигаются МА, что подтверждает его способность обеспечивать стабильное качество решений даже при росте размерности задачи. ГА продемонстрировал сопоставимые медианные показатели, однако он имеет меньшую вариативность результатов, что свидетельствует как о его устойчивости, так и об ограниченном потенциале в достижении экстремально высоких значений. Конфигурации ИО характеризуются значительным разбросом результатов, что указывает на высокую чувствительность к параметрам и необходимость их тщательной настройки. ПА, напротив,

показал более сдержанные результаты, однако при отдельных конфигурациях его эффективность приближается к уровню ГА.

Наибольшее количество задач было наиболее эффективно решено с использованием конкретной конфигурации ПА, что подтверждает его потенциал в качестве специализированного инструмента для ряда сценариев. Наряду с этим, МА демонстрирует наибольшую универсальность: он стабильно достигает высоких результатов при широком диапазоне параметров, тогда как успех ПА обусловлен ограниченным числом конфигураций. ГА и ИО не показали значимого превосходства в общем числе задач, что указывает на их меньшую конкурентоспособность в условиях крупных размерностей, однако данные методы могут быть полезны в роли «быстрых» алгоритмов, обеспечивающих допустимые решения при ограниченных ресурсах времени.

В контексте анализа разброса вычислительного времени и МКС, можно заметить кардинально отличающееся между алгоритмами вычислительное время. Особенно заметна низкая вычислительная сложность МА и ГА, достигающая 15 минут (около 1 млн мс), тогда как ПА и ИО остаются на уровне сотен мс.

У всех алгоритмов, кроме МА ярко заметно слабый прирост МКС со временем, вследствие чего можно сделать вывод, что ГА, ИО и ПА при больших размерностях полезны в качестве «быстрых» алгоритмов. На протяжении процесса решения, МА получает и сохраняет высокий результат, хотя и работает кратно медленнее других. В контексте МКС, высокие результаты сохраняются за МА и ГА, тогда как ПА и ИО начинают заметно уступать им.

В контексте анализа размаха МКС, в сравнении с исследованием ярко заметно превосходство в МКС у некоторых конфигураций МА. Размах МКС у конфигураций ПА и ГА не имеет существенных изменений, у конфигураций ИО он наиболее существенен, чем при задачах меньшей размерности.

Дальнейшая работа велась с 30 конфигурациями ЭА (таблица 3.6).

Таблица 3.6 – выбранные конфигурации эволюционных алгоритмов

МА					
№	Чис-ть акторов	α	β	Q	P
1	50	1,0	2,0	100	0.5
2	50	1,0	2,0	100	0.2
3	50	1.0	5.0	100	0.5
4	50	2,0	5,0	100	0.5
5	50	1,0	2,0	100	0.5
6	50	2,0	2,0	100	0.7
7	50	1,0	1,5	100	0.1
8	50	2,0	5,0	100	0.5
9	20	1,5	0.5	100	0,3
10	20	1,5	0.5	100	0,3
11	20	1,25	0.5	100	0,5
12	20	1,0	1,0	100	0,5
ГА					
№	Чис-ть акторов	Метод отбора		Шанс мутации	Метод мутации
13	100	Турнирный		0.05	СИ
14	100	Турнирный		0.05	РП
15	100	Турнирный		0.01	СИ
16	100	Лучшие N		0.05	СИ
17	200	Турнирный		0.05	СИ
18	50	Рулетка		0.05	СИ
19	50	Турнирный		0.01	СИ
20	50	Турнирный		0.05	РП
ИО					
№	Скорость охлаждения	Температура начальная		Температура конечная	Метод мутации
21	0.95	500		1	СИ
22	0.99	1000		1	СИ
23	0.95	1000		1	СИ
24	0.95	1000		1	СИ
25	0.95	1000		1	РП
26	0.99	500		1	РП
ПА					
№	Чис-ть акторов	Доля пчел-рабочих			Метод мутации
27	50	0.5			СИ
28	20	0.5			СИ
29	20	0.5			СИ
30	20	0.5			РП

Далее, для генерации обучающих данных, проведено множество прогонов [88]: прогнаны уникальные постановки задачи, представляющие собой уникальное сочетание из матрицы смежности, 3 правила с 5 случайными условиями, 10 наборов из 5-15 случайных конфигураций алгоритмов. Для этого сгенерировано 500 тыс. матриц смежности. В общей сложности прогнано 15

млн постановок задач, среди которых отобрано 6 млн, достигших МКС в 0,8 и выше.

Схема алгоритма проведения прогонов показана на рисунке 3.43.

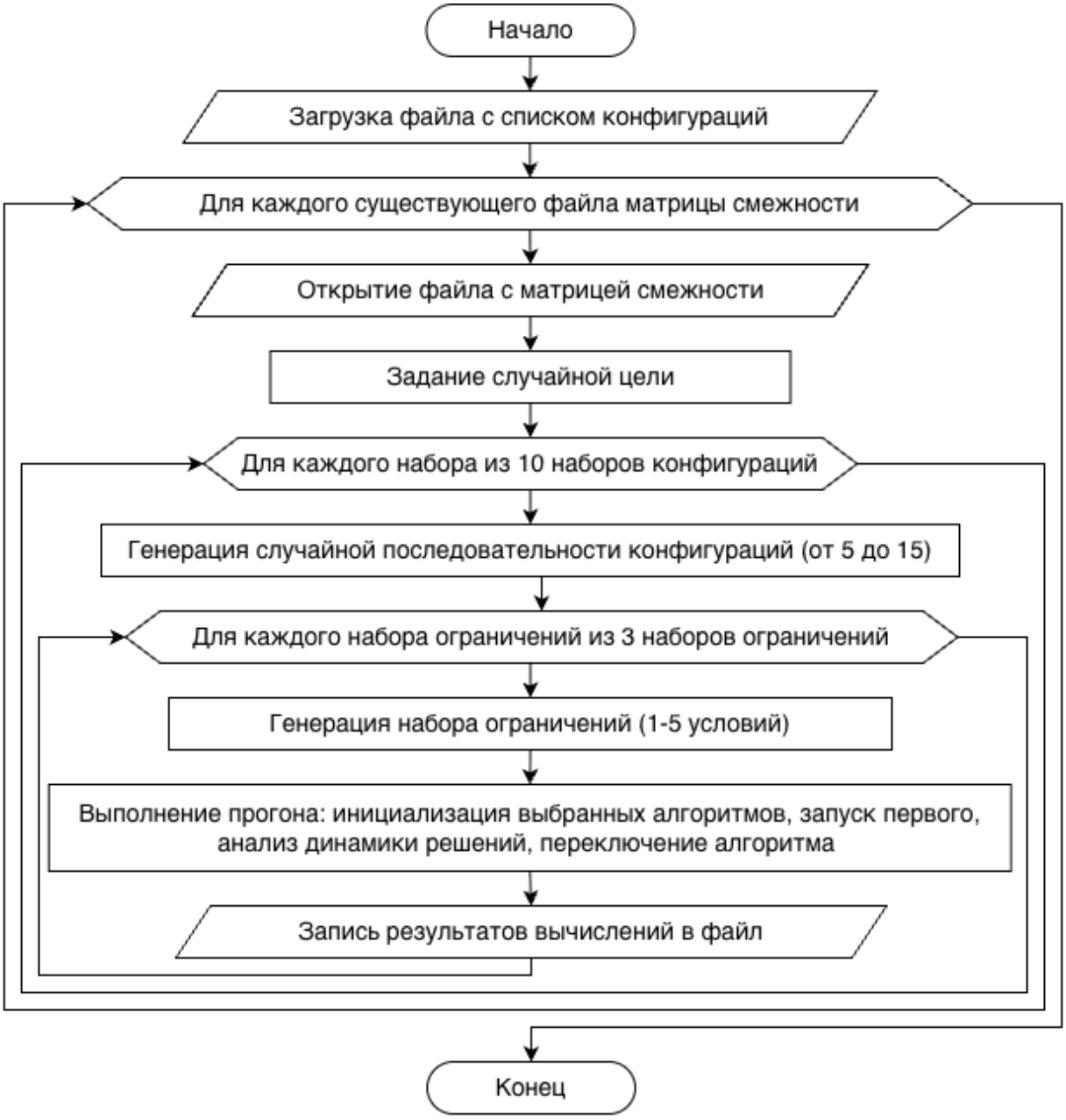


Рисунок 3.43 – Схема проведения прогонов

На начальном этапе осуществляется загрузка входного файла, содержащего перечень доступных конфигураций алгоритмов. Далее инициируется итеративная обработка набора существующих файлов с матрицами смежности. Для каждого такого файла производится его открытие, после чего

формируется случайным образом целевая функция, определяющая критерий оптимизации в данном прогоне.

Для каждого теста создаётся серия из десяти независимых наборов конфигураций. В пределах каждой серии генерируется случайная последовательность конфигураций, используемых при последующем динамическом переключении алгоритмов; длина последовательности варьирует в установленном диапазоне и определяется случайным образом. Параллельно для каждого из трёх наборов ограничений инициируется генерация конкретного поднабора ограничений, включающего от одной до пяти логических условий.

Основной вычислительный этап включает запуск и инициализацию выбранных эволюционных алгоритмов, выполнение первого алгоритма, последующий анализ динамики решений и принятие управляющего решения о переключении на другой алгоритм в соответствии с предопределёнными правилами. После завершения очередного прогона выполняется фиксация результатов, включающая запись полученных решений и служебной информации в выходной файл.

Схема общего алгоритма работы механизма переключения алгоритмов приведена на рисунке 3.44. Модуль переключения ЭА обеспечивает адаптивное управление вычислительным процессом путем последовательного перебора конфигураций, анализа их эффективности и прерывания работы малоэффективных конфигураций. Процесс инициируется получением перечня конфигураций эволюционных алгоритмов, включающих параметры, влияющие на характер поиска: размер популяции, стратегии мутаций и скрещивания, особенности локальных операторов и пр. После получения списка конфигураций запускается внешний цикл обработки, в рамках которого каждая конфигурация поочередно передается в модуль исполнения.

На этапе подготовки конфигурации выполняется инициализация соответствующего класса алгоритма, при которой создается популяция акторов и задаются параметры вычислительного процесса. Если рассматриваемая конфигурация является первой в цикле, алгоритм самостоятельно создает

популяцию. В обратном случае, доля популяции формируется на основе лучших найденных решений предыдущего алгоритма, что обеспечивает преемственность поиска и уменьшает влияние случайности начальных условий.

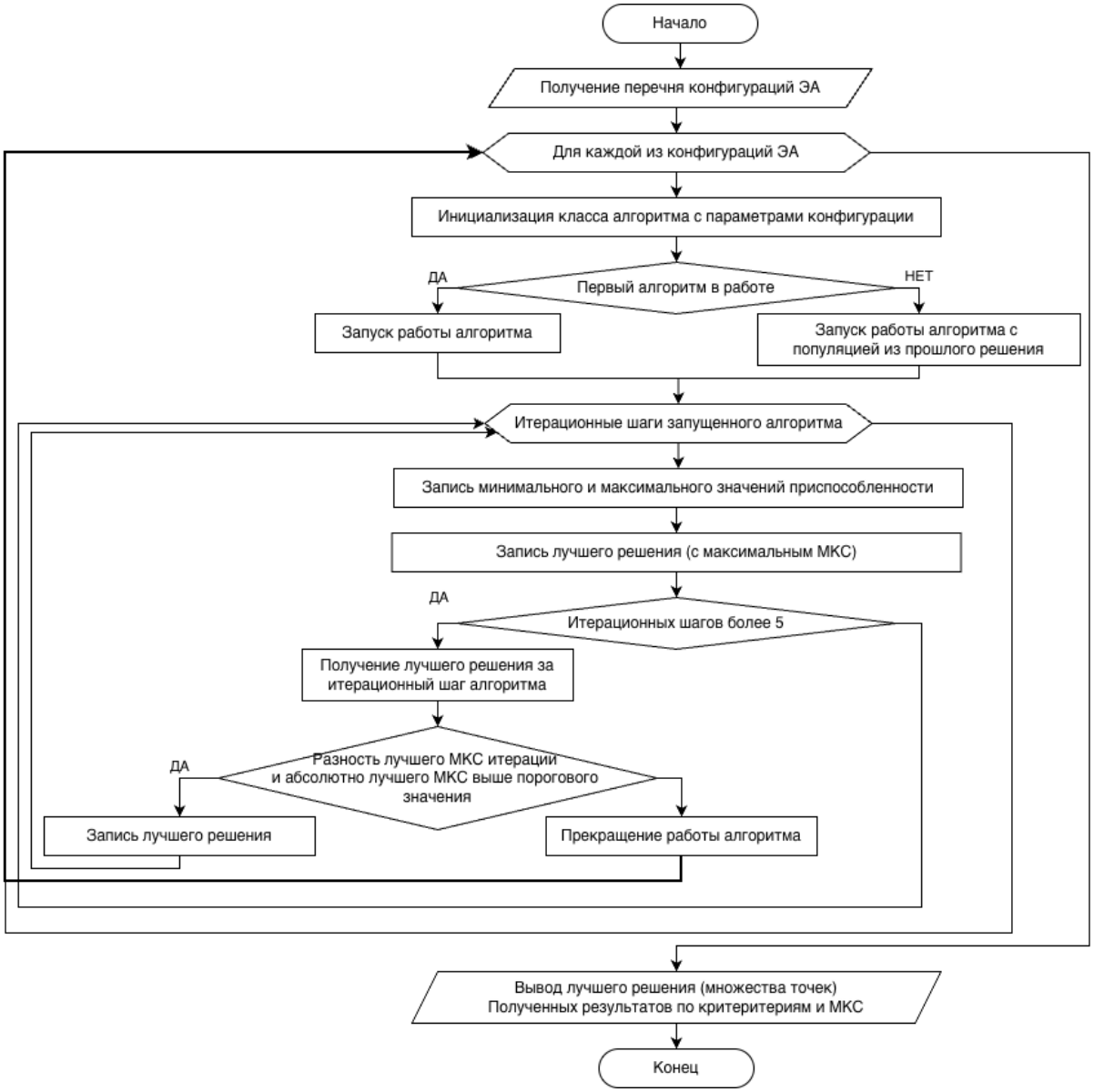


Рисунок 3.44 – Схема общего алгоритма работы переключения алгоритмов

После запуска алгоритма начинается этап итерационной эволюции, который представляет собой последовательность обновлений популяции акторов. На каждой итерации фиксируются минимальное и максимальное значения приспособленности, что позволяет максимизировать корректность вычисления МКС. Параллельно фиксируется лучшее решение, характеризующееся максимальным значением МКС по состоянию текущей итерации.

После выполнения не менее пяти итерационных шагов система получает текущее лучшее решение и сравнивает его МКС с абсолютным максимумом, накопленным в ходе работы всех ранее запущенных конфигураций. Если разность между этими двумя показателями выше порогового значения, вычислительный процесс продолжается. В противном случае выполнение конфигурации прекращается, так как ее дальнейшее использование не обладает практической эффективностью и лишь увеличивает вычислительные затраты.

При обнаружении улучшения глобального максимума МКС текущее лучшее решение записывается как потенциально оптимальное. После завершения работы всех конфигураций или иного условия остановки, модуль выводит лучшее решение, его значение МКС и приспособленности по критериям.

3.6.2 Разработка интеллектуальной модели вариативного выбора конфигураций эволюционных алгоритмов

На вход интеллектуальная модель получает текст ограничений, который в дальнейшем проходит через ряд конвейерных операций для принятия решений.

Первым этапом, специальные операторы заменяются на токены в соответствии с таблицей 3.7.

Таблица 3.7 – токенизация операторов формализации ограничений

Название оператора	Обозначение оператора	Токен	Ид. номер токена
Значение критерия между точками	@	[FROM_TO]	1001
Позиция точки по номеру	#	[ORDER]	1002
Следование точки за точкой	>>	[FOLLOWS]	1003
Следование точки непосредственно за точкой	>>>	[FOLLOWS_IMM]	1004

Продолжение таблицы 3.7

Название оператора	Обозначение оператора	Токен	Ид. номер токена
Диапазон	~	[RANGE]	1005
Логическое «И»	&	[AND]	1101
Логическое «ИЛИ»	|	[OR]	1102
Логическое «НЕ»	!	[NOT]	1103
Равенство	=	[EQ]	1201
Неравенство	!=	[NE]	1202
Меньше	<	[LT]	1203
Меньше или равно	<=	[LE]	1204
Больше	>	[GT]	1205
Больше или равно	>=	[GE]	1206
Разделение выражения и штрафа	->	[ARROW]	1301
Штраф: исключение	exc	[EXCLUDE]	1302
Штраф		[SANCTION]	1303
Разделитель штрафов	;	[SEP_SANCTION]	1304
Служебные токены трансформера			
Разделитель правил		[RULE_SEP]	1305
Начало последовательности		[CLS]	101
Конец последовательности		[SEP]	102
Неизвестный оператор		[UNK]	100
Пропуск		[PAD]	0

В результате предварительной токенизации, выражение

@ 1 2 дистанция > 100 -> exc
 # 3 < 5 -> время:10
>> 1 2 -> exc

преобразуется в выражение следующего вида:

[CLS] [FROM_TO] 1 2 дистанция [GT] 100 [ARROW] [EXCLUDE] [RULE_SEP] [ORDER] 3 [LT] 5 [ARROW] [SANCTION] время:10 [RULE_SEP] [FOLLOWS] 1 2 [ARROW] [EXCLUDE] [SEP]

После выполнения этапа предварительной токенизации исходное текстовое описание ограничений преобразуется в линейную последовательность токенов фиксированного алфавита. При этом сохраняется исходный порядок следования операторов, числовых параметров и идентификаторов точек, что позволяет представить правило в виде формальной структуры, пригодной для дальнейшей машинной обработки.

Следующим этапом является преобразование токенизированной последовательности в числовое представление, используемое в качестве входных данных интеллектуальной модели. Для этого каждому токену сопоставляется уникальный числовой идентификатор, также определенный в таблице 3.7. Числовые значения (идентификаторы точек, значения операндов, параметры штрафов) передаются в модель без изменений или нормализуются в зависимости от типа признака.

Таким образом, результирующая последовательность представляется в виде вектора целых чисел фиксированной длины. Для обеспечения одинакового размера входных данных используется механизм дополнений последовательностей токеном «Пропуск» (PAD), а также маскирование незначимых позиций, не участвующих в вычислении функции внимания трансформера.

Для сохранения информации о положении элементов в выражении дополнительно формируются позиционные признаки, однозначно определяющие индекс токена в последовательности. Это позволяет трансформеру учитывать структуру логических выражений, взаимное расположение операторов, аргументов и санкций, а также корректно интерпретировать вложенные и составные условия.

Итоговое представление входных данных формируется путем «суммирования» эмбеддингов токенов, числовых параметров и позиционных меток.

Полученный набор векторов поступает на вход трансформера, где в рамках механизма «многоголового» внимания выполняется выявление скрытых зависимостей между элементами ограничений, используемое для принятия управляющих решений о выборе и динамическом переключении конфигураций эволюционных алгоритмов в процессе решения задачи.

В результате преобразования, образуется матрица, подходящая для подачи на вход модели. Пример такой матрицы (фрагмент) приведен на таблице 3.8.

Таблица 3.8 – Фрагмент входной последовательности для ИИ-модели

Вектор				Токен
0,12	0,04	-0,08	0,15	[CLS]
0,32	-0,19	0,05	0,17	[FROM_TO]
0,10	0,07	0,01	-0,04	1
0,11	0,11	0,00	-0,04	2
0,48	0,45	-0,10	0,23	дистанция
0,35	-0,23	0,39	-0,10	>
0,81	0,28	0,00	-0,04	100
0,14	0,64	-0,34	0,03	->
0,99	0,09	0,46	-0,41	exc

Матрица эмбеддингов сама по себе не учитывает контекстное влияние элементов друг на друга. Значения порогов, типы штрафов и идентификаторы точек маршрута интерпретируются корректно только вместе с логическими операторами, критериями и структурой правила. Для выявления таких зависимостей и учёта контекста применяется механизм внимания — ключевой компонент трансформерной модели.

Механизм внимания обеспечивает взвешенное агрегирование информации по всей последовательности, позволяя каждому токену учитывать влияние остальных. Это даёт модели возможность устанавливать семантические связи

между операторами ограничений, числовыми параметрами и элементами штрафов независимо от их положения. Особенно важно это при обработке сложных выражений с логическими операторами, вложенными условиями и несколькими правилами.

В используемой трансформерной архитектуре применяется многоголовое внимание, где параллельно вычисляются несколько независимых представлений. Каждая «голова» специализируется на определённом типе зависимостей, например:

– связь операторов сравнения с числовыми порогами;

– соотношение идентификаторов точек и операторов их расположения;

– влияние санкций и исключений на интерпретацию ограничений;

– структурные связи между правилами.

Это позволяет формировать более устойчивое и информативное представление данных по сравнению с одиночным механизмом внимания.

Результатом слоя внимания становится новая матрица признаков, где каждый вектор содержит агрегированную информацию о значимых элементах последовательности. Такое контекстно-зависимое представление используется далее для формирования управляющих решений по выбору и переключению конфигураций эволюционных алгоритмов.

В отличие от рекуррентных моделей, внимание учитывает взаимное влияние токенов независимо от позиции, что критично для корректной интерпретации логических выражений со составными условиями и штрафами.

На вход внимания подаётся матрица эмбеддингов $L \times d$ (L — длина последовательности, d — размерность). Для каждой позиции формируются векторы Query, Key и Value путём линейных преобразований. Механизм вычисляет значимость токенов друг для друга, создавая взвешенное представление с большим вкладом семантически связанных элементов. Например, при обработке штрафа модель учитывает связанное логическое условие, критерий и параметры.

Результаты отдельных «голов» конкатенируются и преобразуются линейно, формируя единую матрицу контекстных признаков той же размерности. Она содержит информацию о структуре, семантике и взаимосвязях элементов ограничений.

Выход многоголового внимания — контекстно-обогащённая матрица, где каждый элемент кодирует своё значение плюс данные о значимых связанных токенах. Это представление передаётся дальнейшим слоям трансформера (нормализация, нелинейности, агрегация). Вектор токена “[CLS]” служит интегральным представлением всех ограничений и входом для модуля принятия решений.

На его основе модель оценивает предпочтительность конфигураций ЭА и решает об их использовании. Таким образом, многоголовое внимание обеспечивает связь между формализованными ограничениями и адаптивным управлением вычислительным процессом.

Выходной слой — полносвязное преобразование, сопоставляющее векторное представление с множеством конфигураций ЭА (каждая имеет уникальный идентификатор и набор параметров: тип метода, популяция, мутации, отбор и др.). В отличие от стандартной классификации, модель выдаёт не одну конфигурацию, а упорядоченный набор идентификаторов, отражающий предпочтительную последовательность применения. Выход — вектор оценок целесообразности каждой конфигурации.

По этим оценкам ранжируются конфигурации, формируя последовательность, учитывающую структуру ограничений, жёсткость, исключения, санкции и логические зависимости. Эта последовательность передаётся внешнему модулю управления, который динамически переключает алгоритмы. Интеллектуальная модель выполняет только стратегическое планирование, не участвуя в оптимизации напрямую.

Внешний модуль (рисунок 3.45) активирует рекомендованные конфигурации, прерывая их по динамике приспособленности и МКС. Образуется двухуровневая архитектура:

– интеллектуальная модель определяет стратегию вариативного использования алгоритмов;

– вычислительный модуль реализует тактическое переключение и контроль.

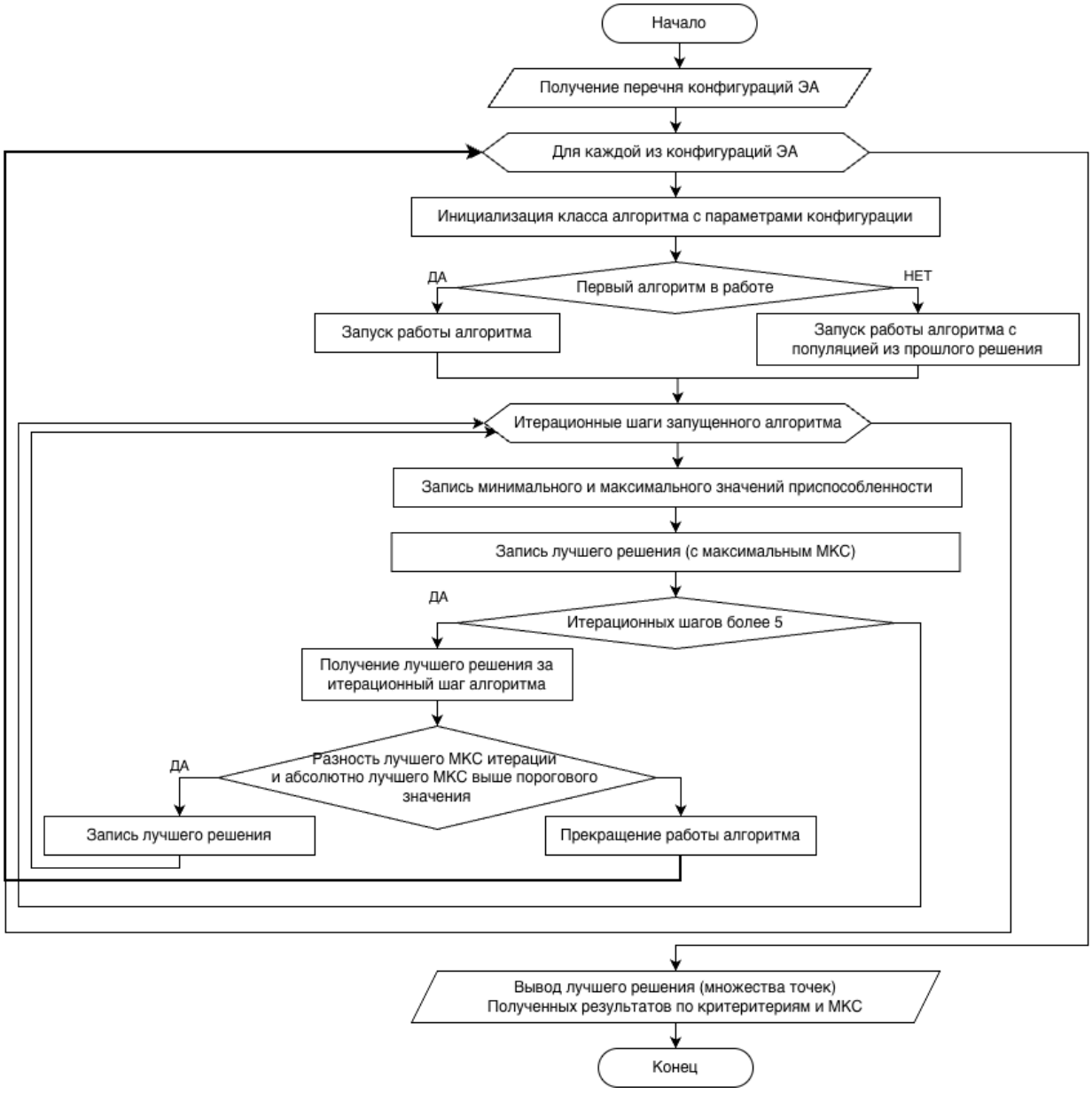


Рисунок 3.45 – Схема алгоритма переключения конфигураций эволюционных алгоритмов

Такое разделение снижает затраты, избегает неэффективных конфигураций и обеспечивает адаптивность при разных структурах ограничений. Выходной слой таким образом связывает формализованное описание задачи с

механизмом вариативного управления эволюционным поиском. Схема модели интеллектуального выбора конфигураций ЭА приведена на рисунке 3.46.

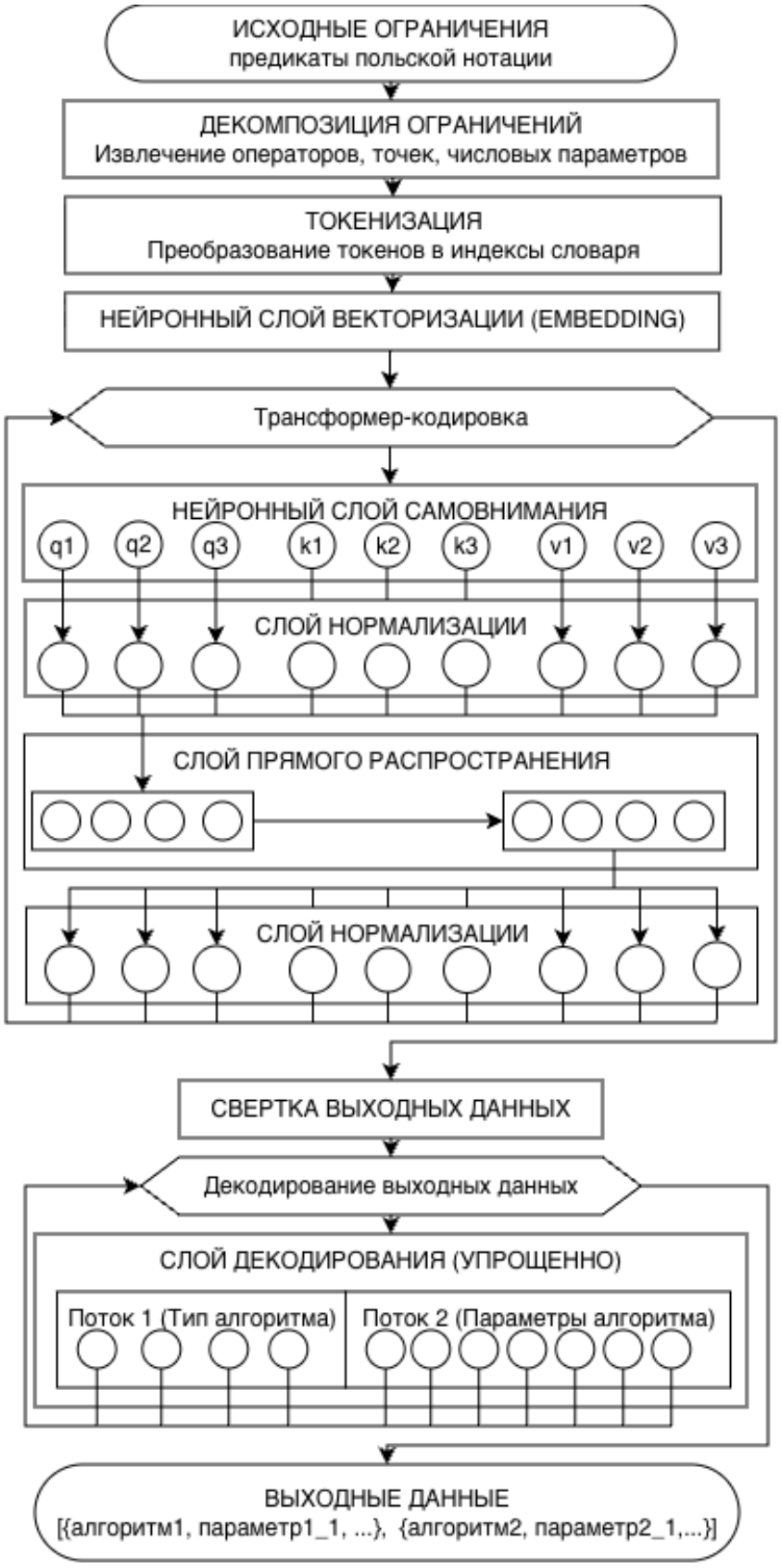


Рисунок 3.46 – Модель интеллектуального выбора конфигураций

В качестве основной метрики оценки качества обучения модели выбрана метрика МКС. Данный выбор обусловлен спецификой решаемой задачи, связанной с многокритериальностью и наличием сложности оценки качества обучения модели, подразумевающий выход сразу нескольких значений. На рисунке 3.47 приведен график обучения интеллектуальной модели. В результате, обученная модель прогнозирует стратегии, способные привести решение задач ЦУО к медианному значению МКС в 0,75 (макс. 1,0).

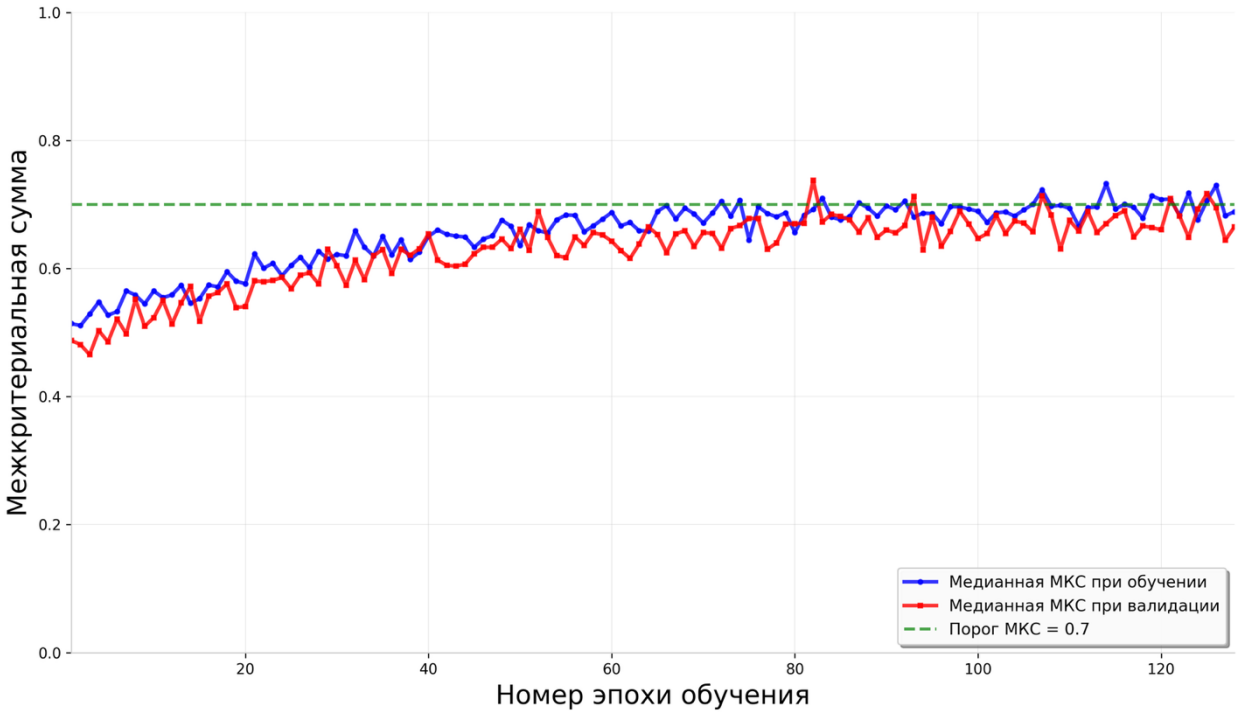


Рисунок 3.47 – График обучения модели

График сравнения интеллектуального решения задачи ЦУО и неинтеллектуального (случайный выбор конфигураций) показан на рисунке 3.48.

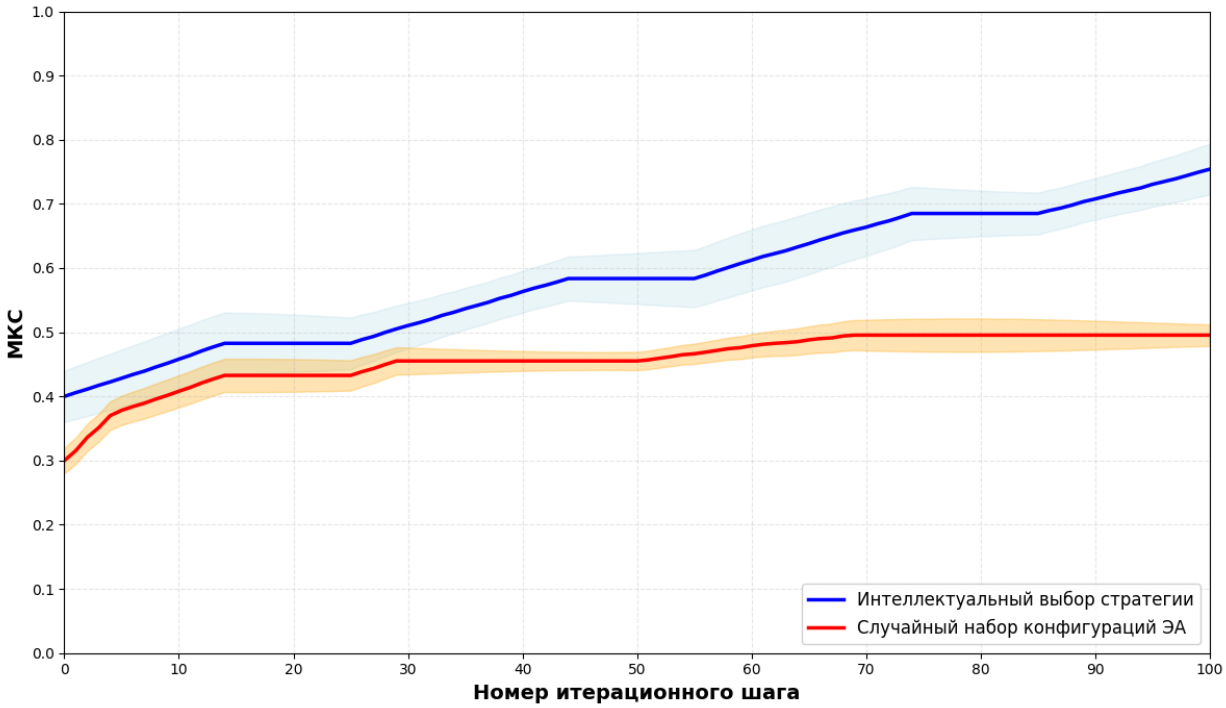


Рисунок 3.48 – Сравнение эффективности интеллектуального выбора стратегии и случайного выбора конфигураций эволюционных алгоритмов

На начальном этапе (итерационные шаги 1-20) оба подхода демонстрируют схожую динамику роста МКС, что свидетельствует о периоде активного поиска решений. В средней фазе (шаги 20-70) интеллектуальный выбор стратегии показывает устойчивый рост качества решения, в то время как случайный подход достигает плато на уровне МКС 0,48–0,5, что указывает на стагнацию вычислительного процесса. На завершающих итерациях, разрыв между подходами достигает максимального значения: интеллектуальный подход достигает МКС в 0,8, тогда как случайный выбор не дает существенного прироста результатов.

Таким образом, применение интеллектуального модуля выбора стратегии обеспечивает прирост качества решений до 50% по сравнению с «случайным» подходом, что подтверждает эффективность предложенного метода интеллектуального формирования последовательности конфигураций эволюционных алгоритмов на основе анализа структуры ограничений задачи.

3.7 Выводы

В данном разделе представлен процесс разработки системы для целочисленной оптимизации с приведением следующих результатов:

1. Созданы средства формализации представления ограничений, позволяющие описывать правила в виде предикатов произвольной степени вложенности и интеграцию штрафных функций в работу эволюционных алгоритмов. Разработанные средства формализации представления ограничений и интерпретатор обеспечивают корректную интерпретацию условий задачи с учетом контекста вычислений.

2. Разработана структура системы принятия решений в условиях многокритериальных ограничений, включая механизм применения штрафов по каждому критерию. Внедрение в систему управления цепочками поставок позволило снизить долю недопустимых решений более чем на 70% по сравнению с базовыми реализациями.

3. Разработан модуль верификации решения задачи целочисленной условной оптимизации, который преобразует выражения, описанные на правилах формирования представления ограничений в логическую функцию верификации решения. В ходе работы эволюционного алгоритма, получаемые решения верифицируются, к решениям, не прошедшим верификацию применяются штрафы к значению приспособленности или полное исключение решения из дальнейших расчетов;

4. Реализованы модифицированные варианты эволюционных алгоритмов с адаптацией под задачи целочисленной условной оптимизации;

5. Разработана интеллектуальная модель вариативного выбора конфигураций эволюционных алгоритмов на основе формализованного представления ограничений задачи с использованием трансформерной архитектуры, что превращает систему целочисленной условной оптимизации в адаптивный искусственный интеллект прикладного уровня. Система получила название АИС ЦУО (адаптивная интеллектуальная система целочисленной оптимизации);

6. На основе проведенных вычислительных экспериментов разной размерности задач отобран перечень используемых конфигураций эволюционных алгоритмов и обучена интеллектуальная модель их выбора для решения задач целочисленной условной оптимизации.

7. Обучена интеллектуальная модель выбора конфигураций эволюционного алгоритма для решения задачи целочисленной условной оптимизации. Эволюционные стратегии, предсказываемые моделью, по медиане обеспечили прирост качества решений до 50% (по сравнению с «случайным» подходом), а значения межкритериальной суммы достигают значения в 0,75.

ГЛАВА 4. АПРОБАЦИЯ РАБОТЫ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ДЛЯ РЕШЕНИЯ РАЗЛИЧНЫХ ЗАДАЧ ЦЕЛОЧИСЛЕННОЙ УСЛОВНОЙ ОПТИМИЗАЦИИ

4.1 Целочисленная условная оптимизация в логистике и транспортных системах

Разработанная система (адаптивная интеллектуальная система целочисленной оптимизации – АИС ЦУО) может быть успешно применена, в первую очередь, в логистических задачах, направленных на поиск кратчайшего пути с учетом нескольких критериев. Логические процессы характеризуются множеством взаимосвязанных факторов, таких как время доставки, стоимость перевозки, загруженность транспортных сетей и т.д.

В частности, под логистическими задачами, в качестве примера, можно привести следующие:

1. маршрутизация транспорта, для которой нужно сократить время, затраты, но максимизировать надежность и важность;
2. управление цепочками поставок, где необходимо координировать работы складов, транспорта и точек выдачи. Здесь критериями выступает учет временных окон, минимизация простоев, максимизация скорости обработки заказов и др.;
3. планирование авиаперелетов и железнодорожных перевозок. Основную роль, здесь выступает планирование расписаний и маршрутов с учетом множества факторов, которые будут учитывать загруженность станций, минимизация расхода топлива, использования транспортных средств и персонала и т.д.

В таблице 4.1 приведена постановка задачи маршрутизации транспорта с 4 критериями: дистанция, время, важность, пропускная способность.

Таблица 4.1 – Постановка задачи маршрутизации транспорта

Дистанция (минимизация)									
0	78	141	175	207	209	246	278	315	356
78	0	82	106	139	135	169	200	237	280
141	82	0	39	68	78	120	155	195	229
175	106	39	0	33	39	81	116	156	189
207	139	68	33	0	29	65	98	137	165
209	135	78	39	29	0	42	77	117	151
246	169	120	81	65	42	0	35	75	111
278	200	155	116	98	77	35	0	40	79
315	237	195	156	137	117	75	40	0	49
356	280	229	189	165	151	111	79	49	0
Время (минимизация)									
0	1192	1817	6525	7150	8342	8967	1375	707	5818
1192	0	625	5333	5958	7150	7775	2551	637	4628
1817	625	0	4708	5333	6525	7150	3172	1222	4005
6525	5333	4708	0	625	1817	2442	7871	5898	779
7150	5958	5333	625	0	1192	1817	8496	6522	1375
8342	7150	6525	1817	1192	0	625	9687	7713	2551
8967	7775	7150	2442	1817	625	0	101312	8338	3172
1375	2551	3172	7871	8496	9687	10312	0	1975	7150
707	637	1222	5898	6522	7713	8338	1975	0	5175
5818	4628	4005	779	1375	2551	3172	7150	5175	0
Важность (максимизация)									
0	4000	6000	8000	12166	2828	2828	6325	8426	10770
4000	0	2000	4000	16125	6325	2828	2828	4472	14560
6000	2000	0	2000	18111	8246	4472	2000	2828	16492
8000	4000	2000	0	20100	10198	6325	2828	2000	18439
12166	16125	18111	20100	0	10000	14000	18000	20000	2828
2828	6325	8246	10198	10000	0	4000	8000	10000	8246
2828	2828	4472	6325	14000	4000	0	4000	6000	12166
6325	2828	2000	2828	18000	8000	4000	0	2000	16125
8246	4472	2828	2000	20000	10000	6000	2000	0	18111
10770	14560	16492	18439	2828	8246	12166	16125	18111	0

Продолжение таблицы 4.1

Пропускная способность (максимизация)									
0	32	12	56	78	23	45	67	89	10
34	0	23	45	67	89	11	32	54	65
12	23	0	14	35	56	78	20	41	62
56	45	14	0	15	35	57	79	21	43
78	67	35	15	0	16	37	58	79	22
23	89	56	36	16	0	17	38	59	80
45	11	78	57	37	17	0	18	39	60
67	32	20	79	58	38	18	0	19	40
89	54	41	21	79	59	39	19	0	24
10	65	62	43	22	80	60	40	24	0

Два из приведенных критериев имеют цели «минимизация» и два – «максимимизация». Каждый из приведенных критериев находится в разных числовых диапазонах. Далее, он будет подан на вход разработанной системе. График прогресса решения поставленной задачи показан на рисунке 4.1.

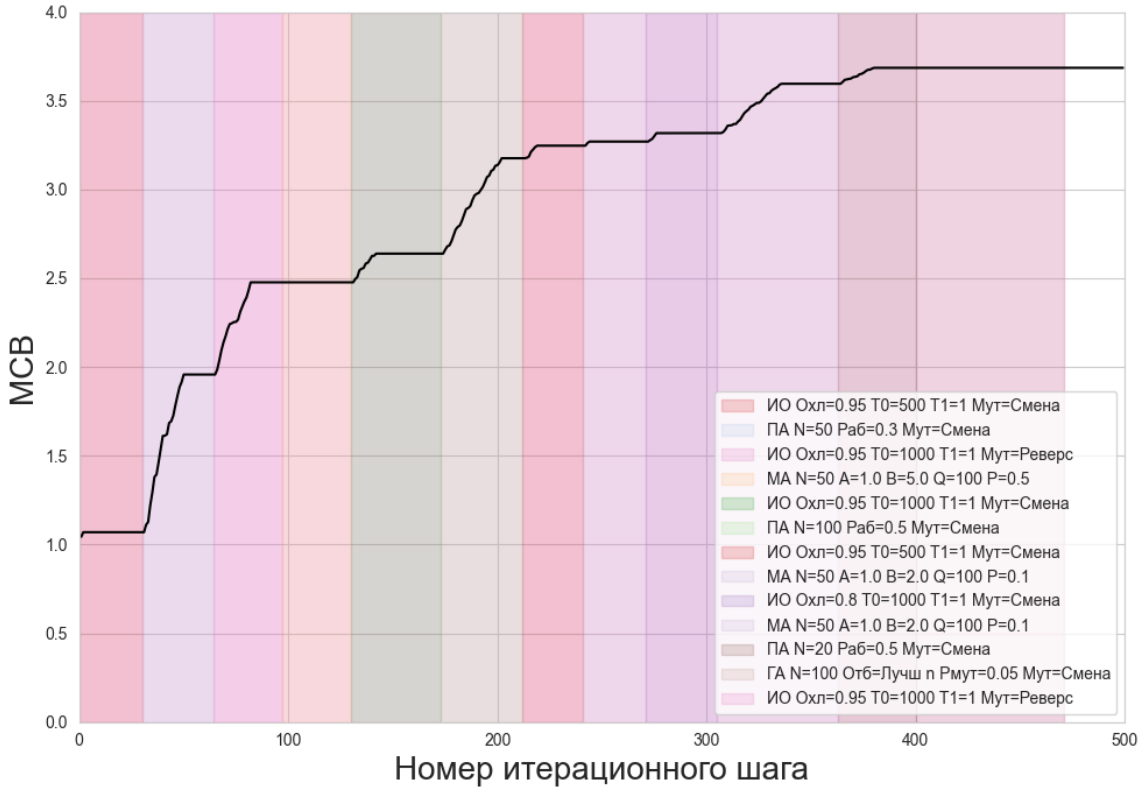


Рисунок 4.1 – График прогресса решения поставленной задачи

В начале решения наблюдается ожидаемый резкий рост МКС, однако, по мере увеличения числа итераций, рост замедлялся и на тех моментах, когда улучшения не происходило, изменялся конфигурация алгоритма изменялась в соответствии с прогнозами, полученными от ИИ-модели. Зачастую, это приводило к новому скачку в прогрессе решения.

В начале, в силу своей быстроты, использовались конфигурации ИО, однако они не дали ожидаемой сходимости, вследствие чего решение было передано ПА. Частое замедление видно на поздних этапах решения: конфигурации МА и ПА давали небольшой рост, и системе приходилось переключать методы как на быстрые, так и на качественные для улучшения результатов. Решение остановилось на отметке МКС в 0,921.

Далее, для этой задачи были выставлены ограничения. Небольшое число правил (2-3) не внесло существенные изменения в прогресс решения, поэтому было составлено большее число ограничений, наиболее близких к реальным промышленным задачам. Ограничения, примененных к задаче, приведены на таблице 4.2.

Таблица 4.2 Примененные к задаче ограничения

№	Текст ограничения	Штраф
1	@ 0 1 дистанция > 300 & >> 1 3 & # 3 ~ [2,5]	время:200 важность:-10
2	(@ 2 4 время < 1500 & @ 4 6 важность < 20)	исключение
3	# 0 = 0 & # 9 = 9 && (>> 5 2 | >>> 8 7) & @ 5 8 пс >= 5000	пс:-200 время:500
4	# 1 ~ [0,2] & (!(@ 1 2 важность > 40) | @ 1 3 время > 2000)	важность:-30 время:1000
5	(@ 7 8 пс < 582 & # 7 > # 8)	исключение
6	!(@ 3 5 важность >= 50 & @ 5 7 время <= 1000)	важность:-40 время: 1200
7	# 9 ~ [8,9] & @ 8 9 пс < 350	время:2000 важность: -20
8	@ 1 2 дистанция > 50 & # 4 ~ [2,7] | @ 3 4 время <= 100 & # B = 0	исключение
9	!(>> 3 5 | >>> 6 8) & @ 4 3 дистанция != 35	важность:-15 дистанция:500

Ниже приведено краткое описание причин выбора данных ограничений:

1. некоторые ограничения позволяют сократить пространство поиска, сосредоточившись на реалистичные или приоритетные варианты.

Например, ограничение «@ 0 1 дистанция > 300 -> exc» исключает маршруты с избыточно длинными связями, уменьшая количество вариантов для анализа;

2. санкции вида «важность:-30» отражают ценности заказчика в виде приоритета важности над расстоянием. Или если важность между некоторыми городами должна быть выше определенного значения, то штраф в виде «важность:-40» заставит алгоритмы избегать таких маршрутов;

3. логистическая целесообразность посещения может быть описана при помощи операторов «Следование точки за точкой» и «Следование точки непосредственно за точкой». Например, посещение склада в точке 5 целесообразнее посетить раньше точки 2, в которой находится клиент;

4. ограничения вида «# 0 = 0 & # 9 = 9» зафиксируют старт и финиш решений, что может быть критично в ряде задач, когда, например, в этих точках находится депо.

На рисунке 4.2 приведен график прогресса решения поставленной задачи с ограничениями.

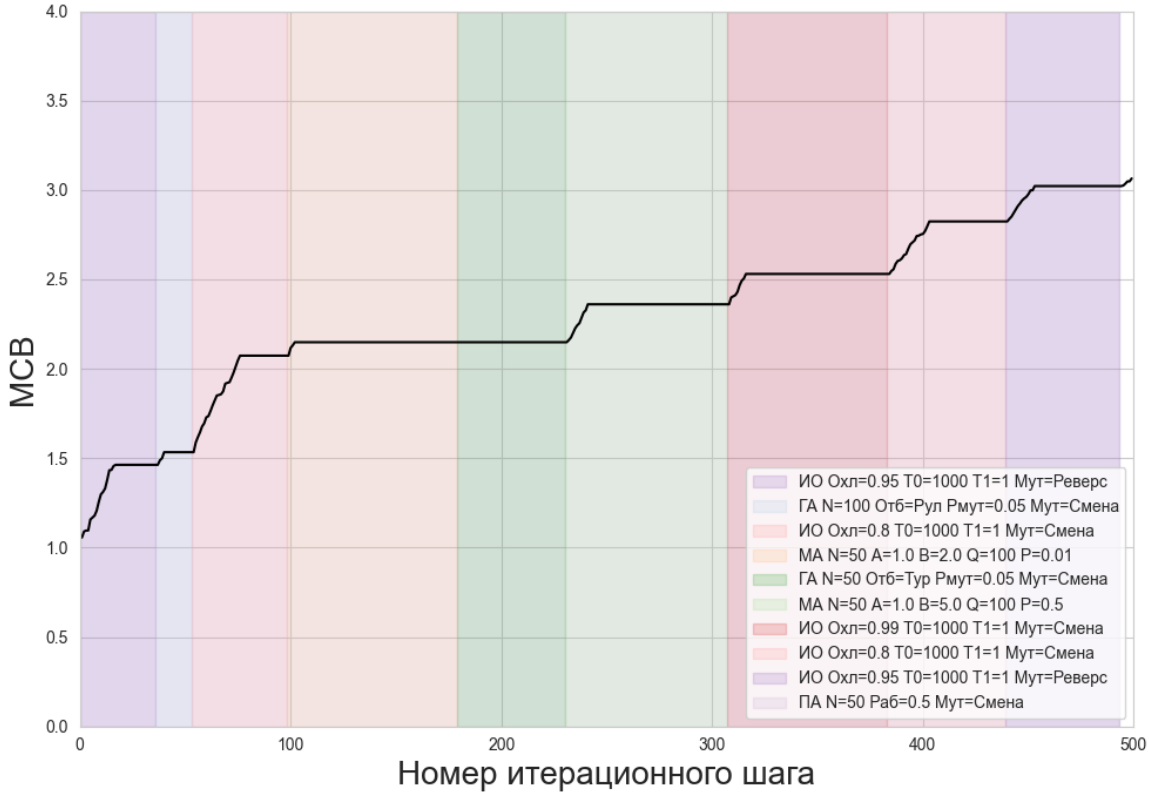


Рисунок 4.2 – График прогресса решения задачи с ограничениями

На втором графике результаты заметно ухудшились: рост МКС медленнее по сравнению с решением без ограничений и достигаются длительные «застои» уже на 100 шаге. Кроме того, алгоритмы реже переключались, что также стало причиной стагнации. Вероятно, большая интеграция ПА в ИИ-модель могло бы дать больший прогресс в решении. Тем не менее, достижение МКС в 3.0 является удовлетворительным и говорит о том, что ИИ-модель имеет смысл дообучить по мере роста объема данных для исследования и дальнейшего обучения, а параметры подвергнуть более детальной регулировке.

4.2 Оптимизация конечных автоматов в управлении сложными системами

Оптимизация конечных автоматов (далее – КА) является ключевой задачей в проектировании эффективных систем управления, компиляторов и искусственного интеллекта [89]. Традиционные методы, какие как минимизация состояний или детерминация, часто фокусируются на одном критерии, например, уменьшении числа состояний. Однако в реальных приложениях требуется учитывать несколько параметров: время переходов, энергопотребление, надежность или стоимость.

Каждый критерий (например, время перехода или вероятность сбоя), можно задать в виде графа и, соответственно, матрицы смежности. Для неориентированных графов матрица симметрична, для ориентированных асимметрична. Например, для КА с n состояниями матрица будет иметь размер $n \times n$ и содержать числовые или логические значения, характеризующие переходы. Кроме того, матрицы смежности адаптируются к любой структуре, включая автоматы с выходными сигналами (Мили-Мура). Таким образом, задача сводится к выбору такой конфигурации автомата и набора переходов, которые обеспечивают сбалансированность по нескольким критериям качества. Для этого применяются методы оптимизации, где ЭА позволяют учитывать одновременно задержки, надежность и стоимость реализации, где ЭА выступают

инструментом адаптации к многокритериальным требованиям реальных систем.

Существует ряд решений, предназначенных для оптимизации конечных автоматов [90, 91, 92, 93]. Все они представляют собой интегрируемое ПО, разработанное в разное время на разных языках программирования. Сравнительный анализ программ-аналогов для оптимизации КА приведен на таблице 4.3.

Таблица 4.3 – Сравнительный анализ существующих программных пакетов для оптимизации конечных автоматов

Наименование ПО	Язык программирования	Основные возможности	Особенности
DFA-Minimizer	Java	Минимизация детерминированных КА, преобразование недетерминированных КА в детерминированные, графическое представление переходов	Подходит для образовательных целей, простота использования
DFA-Minimization	Python	Минимизация КА на основе теоремы эквивалентности	Легковесное ПО для быстрых экспериментов и прототипирования
fsmmin	Python	Анализ и минимизация конечных автоматов Мили, визуализация, применение алгоритма Мура	Интерактивное графическое представление автоматов
Finite-State-Machine-Minimization	Python	Визуализация, выполнение и минимизация конечных автоматов в интерактивном режиме	Интерактивный режим работы с возможностью демонстрации промежуточных результатов

Представленные программные пакеты демонстрируют разнообразие подходов к оптимизации КА с акцентом на минимизацию состояния и улучшение визуальной интерпретации результатов. Программы, реализованные на Java, выделяются своей наглядностью и удобством для образовательных целей. С другой стороны, программные пакеты на Python подходят для быстрого прототипирования и интеграции в скриптовые решения, предлагая интерактивную визуализацию.

В качестве кейса для апробации рассматривается оптимизация управляющего автомата для промышленного устройства реального времени – например, протокола управления роботизированной сборочной линией. Каждый переход автомата соответствует выполнению конкретной команды, обработке сигнала датчика или переключению состояния исполнительного механизма. Фактически, КА описывает последовательность действий системы, где критически важны:

– задержка перехода – задержка между сигналом датчика и выполнением соответствующей команды;

– надежность перехода – вероятность корректного срабатывания механизма без ошибки;

– стоимость перехода – затраты энергии или ресурсов на выполнение операции.

Каждый критерий представлен в виде матрицы смежности, где строки и столбцы соответствуют состояниям автомата, а значения – численные характеристики переходов. Например, для автомата с n состояниями матрица имеет размер n × n и отражает реальные параметры работы системы.

Множество ограничений задает технологические и эксплуатационные требования: некоторые переходы могут быть запрещены, другие должны выполняться строго в определенной последовательности, а суммарная задержка не должна превышать допустимого порога. Таким образом, задача сводится к поиску оптимальной последовательности действий автомата, обеспечивающей баланс между быстродействием, надежностью и ресурсозатратностью.

Для примера будет рассмотрен автомат в соответствии с формулой (4.1).

$$K = (S, \epsilon, \sigma, q_0, F) \tag{4.1}$$

Где $S = \{0,1,2,3\}$ – состояния автомата, отражающие шаги процесса;

$\epsilon = \{A, B, C, D, E, F, G, H, I, J\}$ – входной сигнал датчиков;

$q_0 = A$ – начальное состояние;

$F = \{I, J\}$ – единственное конечное состояние.

Информация о критериях приведена на таблице 4.4.

Таблица 4.4 – Критерии задачи оптимизации конечного автомата

Стоимость переходов (минимизация)										
	A	B	C	D	E	F	G	H	I	J
A	0	3	5	4	6	7	5	8	6	4
B	4	0	6	5	3	4	7	5	6	8
C	5	4	0	7	5	6	3	8	7	5
D	6	5	4	0	7	3	5	6	8	7
E	7	6	5	3	0	4	6	5	7	8
F	5	7	6	4	5	0	7	6	4	5
G	6	5	7	5	4	6	0	7	5	6
H	8	6	5	7	6	5	4	0	6	7
I	7	8	6	5	7	5	6	4	0	5
J	5	7	8	6	5	7	6	5	4	0
Надежность переходов (максимизация)										
	A	B	C	D	E	F	G	H	I	J
A	0	0.9	0.85	0.8	0.88	0.9	0.86	0.92	0.87	0.8
B	0.87	0	0.9	0.83	0.88	0.85	0.9	0.86	0.89	0.84
C	0.9	0.88	0	0.85	0.87	0.9	0.84	0.88	0.85	0.89
D	0.85	0.87	0.88	0	0.9	0.86	0.85	0.9	0.87	0.88
E	0.88	0.85	0.86	0.9	0	0.87	0.88	0.85	0.9	0.86
F	0.9	0.86	0.88	0.87	0.85	0	0.9	0.88	0.86	0.9
G	0.87	0.88	0.85	0.9	0.86	0.87	0	0.85	0.88	0.86
H	0.9	0.87	0.88	0.86	0.85	0.88	0.86	0	0.87	0.88
I	0.86	0.9	0.87	0.88	0.86	0.87	0.85	0.9	0	0.86
J	0.88	0.86	0.9	0.87	0.88	0.85	0.87	0.86	0.88	0
Задержка переходов (минимизация)										
	A	B	C	D	E	F	G	H	I	J
A	0	5	8	6	7	5	9	4	6	7
B	6	0	7	5	8	6	5	7	9	4
C	7	6	0	8	5	7	6	5	8	6
D	5	7	6	0	7	5	8	6	5	7
E	6	5	7	5	0	6	7	5	6	8
F	7	6	5	6	5	0	7	6	5	7
G	8	7	6	5	6	7	0	6	7	5
H	6	5	7	6	5	6	5	0	6	7
I	5	6	5	7	6	5	6	5	0	6
J	7	6	5	6	7	6	5	7	6	0

Перечень правил, описывающих ограничения и примененных к задаче, приведен в таблице 4.5.

Таблица 4.5 – Правила, формализующие ограничения, примененные к поставленной задаче оптимизации конечного автомата

№	Текст ограничения	Санкция
1	>>> B D & >> C F	Стоимость:3 Надежность:0.9 Задержка:5
2	@ A G задержка <= 6 & @ E H задержка ~ [5,7]	Стоимость:4 Надежность:0.85 Задержка:7
3	# A = 1 & # J = 10 & # C ~ [4,6]	Исключение
4	!(>> G B) & @ H F задержка <= 7	Исключение
5	(@ D I задержка > 5 & @ E F задержка < 4)	Стоимость:3 Надежность:10 Задержка:5
6	@ A B надежность < 0.85 & @ C D надежность > 0.9	Исключение
7	# F = 5 & @ F G задержка < 6	Исключение
8	>> A B & @ B C стоимость <= 6	Стоимость: 11
9	>>> E F & @ F H надежность > 0.9	Исключение
10	@ D G задержка ~ [4,8]	Исключение
11	# H ~ [7,9] & >> H I	Стоимость:3 Надежность:0.87 Задержка:6
12	!(# A = 1)	Исключение
13	@ C E задержка < 7 & @ E G надежность > 0.88	Стоимость:3 Надежность:12 Задержка:7
14	# B ~ [2,3] & >>> B D	Стоимость: 2 Надежность:0.9 Время:5
15	@ G H задержка > 5	Надежность:0.85 Задержка:13
16	>> F J & @ F I стоимость < 7	Исключение
17	# E = 4 & @ E F задержка <= 6	Стоимость:3 Надежность:0.88 Задержка:8
18	@ A D надежность < 0.86	Стоимость:2 время:5
19	>>> C F & @ F G задержка ~ [5,7]	Стоимость:3 Надежность:0.9
20	!(# I = 10) | !(# J = 10)	Исключение

Заданный набор ограничений формирует жесткие требования для оптимизации конечного автомата. Каждое ограничение фиксирует либо положение

состояния в решении, либо характеристики конкретного перехода, либо полностью исключает нежелательные решения (например, низкая надежность между определенными состояниями). Это позволяет при поиске оптимального решения учитывать все существенные критерии и гарантировать, что выбранная последовательность будет соответствовать заранее определенным параметрам качества работы автомата.

Для проведения сравнительного анализа рассматриваемых программных пакетов в контексте поставленной задачи, для получения и обработки результатов, были сформированы следующие принципы работы:

– т.к. пакеты «нацелены» на минимизацию, матрица критерия с целью максимизации (Надежность переходов) была подана на вход в инвертированном виде (умноженном на -1);

– т.к. пакеты не предусматривают работу с ограничениями, проведено множество прогонов по всем критериям до тех пор, пока не было получено и отобрано решение, удовлетворяющее всем поставленным ограничениям (валидное решение);

– т.к. пакеты не поддерживают работу с задачами ЦУО, для множества корректных решений было вычислено значение МКС;

– показателем вычислительного времени для каждого программного пакета выступает медианное значение вычислительного времени среди всех прогонов (в пределах используемой программы), выдавших валидные решения.

График результатов сравнительного анализа по МКС и вычислительному времени показан на рисунке 4.3.

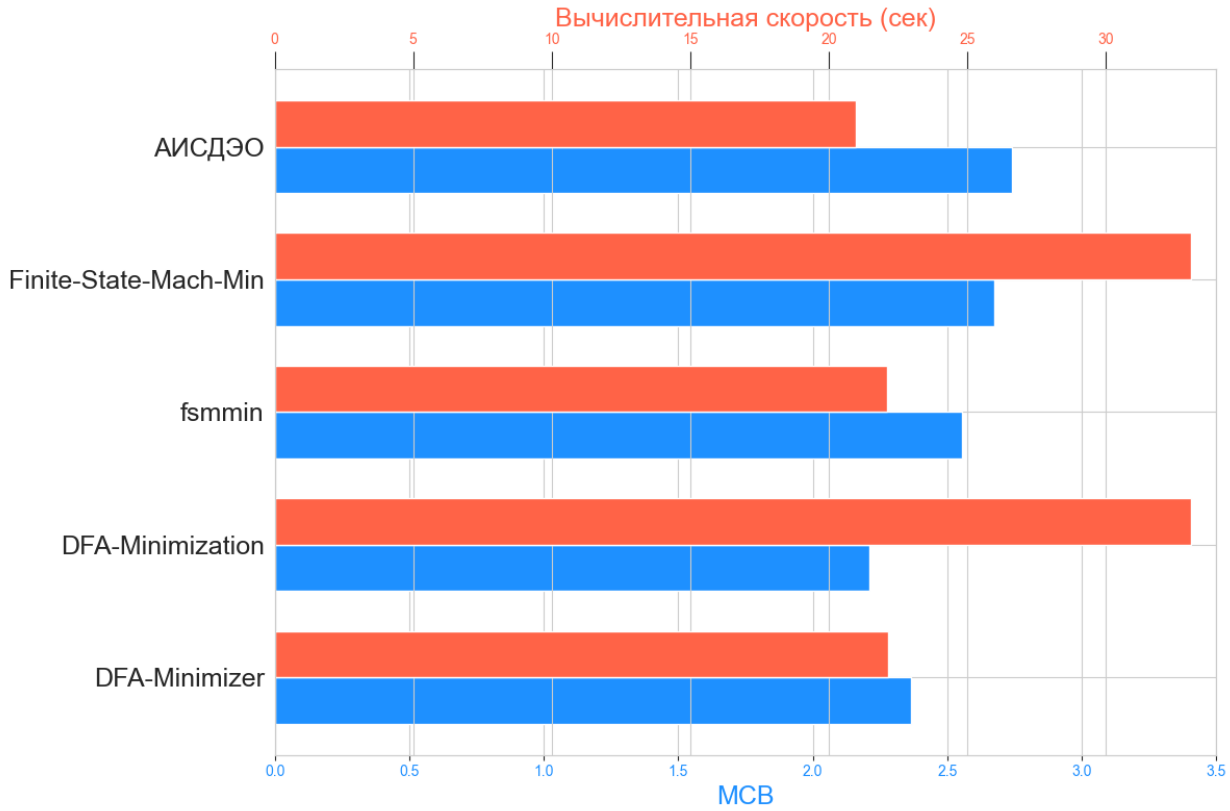


Рисунок 4.3 – График сравнительного анализа программных пакетов для решения задачи оптимизации конечного автомата

По представленному графику можно сделать несколько основных выводов:

– АИСЦУО демонстрирует наибольшее МКС при минимальной вычислительной скорости. Это говорит о том, что данный инструмент обеспечивает лучшую сбалансированность по совокупности критериев и в то же время обрабатывает задачу быстрее остальных решений.

– Finite-State-Machine-Minimization показывает высокие показатели МКС, однако время вычислений у него самое большое. Такой результат указывает на хорошую оптимизацию конечных автоматов по выбранным критериям, но с потерей в производительности.

– fsmmin занимает среднюю позицию по обоим показателям: лучше, чем некоторые инструменты, но уступает лидеру по качеству и скорости.

– DFA-Minimization и DFA-Minimizer показывают результаты ниже среднего по МКС и не самые высокие по скорости. При этом DFA-Minimizer чуть быстрее и немного лучше по МКС, чем DFA-Minimization.

Если учитывать приоритет быстрого времени отклика, то лучшим выбором будет АИСЦУО. Если же приоритет — максимально возможное средневзвешенное качество при допустимом времени выполнения, то Finite-State-Machine-Minimization тоже может быть привлекательным, хотя и работает заметно медленнее.

4.3 Перспективы применения целочисленной оптимизации в иных отраслях

Некоторые другие задачи, которые будут описаны далее, также могут быть представлены в виде ряда критериев со целями и ограничениями. У них ключевыми моментами оптимизации выступают иные факторы. Разработанное программное обеспечение, при внесении ряда модификаций, может быть применено в ряде других направлений и отраслей.

4.3.1 Перспективы применения в теории игр

Теория игр изучает стратегическое взаимодействие агентов в условиях конкуренции или сотрудничества, где выбор каждого игрока влияет на результаты других. Задачи ЦУО в области теории игр возникают, когда участники стремятся оптимизировать разные функции (например, прибыль, риск, репутация). Система АИС ЦУО может быть адаптирована для решения игровых моделей с противоречивыми критериями, что соответствует требованиям современных экономических и управленческих задач.

Разработанная система позволяет каждому игроку задать собственную матрицу смежности и цель (например, максимизация прибыли или минимизация риска). Это напрямую соответствует модели некооперативных игр, где участники действуют независимо. Например, в олигополии компании могут конкурировать за рынок, используя разные стратегии:

– компания А: максимизирует прибыль (матрица с ценами и спросом).

– компания Б: минимизирует риски (матрица с коэффициентами нестабильности рынка).

Разработанный подход с эволюционными алгоритмами может находить равновесия Нэша — стратегии, где ни один игрок не заинтересован менять свою позицию, зная стратегии других [92]. Например, в аукционе с несколькими участниками АИСЦУО определит оптимальные ставки, учитывая ограничения (например, «ставка не должна превышать бюджет»).

Задачи оптимизации в играх требуют учета нескольких параметров (например, прибыль, экологичность, социальная ответственность). АИСЦУО решает это через минмакс-нормализацию критериев и объединение их значений. Это позволяет находить компромиссные решения, где улучшение одного критерия не ухудшает другого (принцип Парето).

Ограничения типа "следует за" или "точка по порядку" могут моделировать логические условия в играх. Например, в игре с неполной информацией игроки могут действовать в определенной последовательности, что соответствует последовательным играм.

Ниже приведены примеры применения целочисленной оптимизации в теории игр [94, 95]:

– в олигополии несколько компаний конкурируют за долю рынка. Система поможет определить оптимальные стратегии ценовой политики, учитывая Прибыль (матрица с ценами и спросом), риски (матрица с вероятностью банкротства), ограничения (например, «цена не может быть ниже себестоимости»). Это соответствует задачам, где теория игр анализирует функционирование экономических систем.

– на аукционе участники могут оценивать лот по нескольким параметрам (стоимость, качество, сроки поставки). АИСЦУО найдет оптимальные ставки, балансирующие эти критерии, что актуально для аукционных моделей;

– В кооперативных играх участники стремятся максимизировать общую прибыль, но при этом могут иметь личные цели (например, "компания А хочет получить 60% прибыли"). Система поможет определить оптимальное распределение ресурсов, учитывая ограничения [96].

4.3.2 Перспективы применения в теории принятия решений и динамических системах

Теория принятия решения (далее – ТПР) занимается разработкой формальных методов для выбора варианта из множества альтернатив, учитывая несколько конкурирующих критериев. Поскольку в задачах ЦУО часто возникают противотечения между целями (например, минимизация затрат против максимизации качества), АИСЦУО может быть адаптировано для динамических систем, где требуется баланс между критериями [97]. Преимуществом использования АИСЦУО в ТПР является способ вычисления конечной приспособленности через минмакс-нормализацию и объединение значений критериев. Минмакс-нормализация уравнивает масштабы критериев, а суммирование нормированных значений позволяет находить компромиссные решения подобно принципу Эджворта-Парето [98].

Разработанное решение способно автоматизировать процесс поиска оптимальных решений, минимизируя субъективность, характерную для традиционных методов, таких как аналитическая иерархия или критерий Лапласа [99]. Например, в задачах распределения ресурсов, алгоритмы могут учитывать не только финансовые ограничения, но и логические условия, такие как «покупка книг по базовым дисциплинам должна предшествовать приобретению специализированных изданий». Это соответствует требованиям, где подчеркивается важность учета иерархии при принятии решений.

В бизнес-сфере система может быть интегрирована в процессы принятия решений, описанные в [100, 102], где ключевыми являются прозрачность и логичность. Например, при выборе стратегии развития компании, где критерии включают прибыльность, экологическую устойчивость и социальную

ответственность, АИСЦУО формирует набор решений, из которых специалист выбирает наиболее подходящий.

Сравнение с традиционными методами подтверждает преимущество эволюционных алгоритмов в условиях неопределенности: они позволяют учитывать вероятностные сценарии и сложные ограничения, что недостижимо для простых математических моделей. Важно обеспечить корректную формализацию критериев, чтобы избежать искажений, характерных для методов, основанных на упрощенных представлениях (например, стемминга в обработке текстов).

Таким образом, разработанная система является инструментом, подходящим в том числе и для решения задач ЦУО в сфере ТПР, где требуется учет противоречивых целей, динамических ограничений и вероятностных факторов, что соответствует требованиям современных научных и практических задач.

4.3.3 Перспективы применения на мобильных устройствах

Архитектура АИС ЦУО позволяет адаптировать систему для работы на устройствах с ограниченными вычислительными ресурсами. Вычислительное ядро системы реализовано на языке программирования Rust, что обеспечивает эффективное управление памятью и минимальные накладные расходы. Это позволяет компилировать систему в том числе и для архитектур ARM и RISC-V, широко используемых в том числе и на мобильных устройствах и одноплатных компьютерах (например, Raspberry PI).

Применение системы на одноплатных компьютерах позволяет создавать автономные интеллектуальные контроллеры промышленного назначения. Описанные выше задачи (но не ограничиваясь ими) могут решаться локально без постоянного соединения с сервером.

Особый интерес представляет интеграция системы в мобильные решения для транспортной логистики. Современные планшетные компьютеры и специализированные терминалы обладают достаточной производительностью

для решения оптимизационных задач средней размерности. Оператор может получать актуализированные маршруты с учётом текущей ситуации, динамически изменяющихся приоритетов доставки и временных ограничений. При этом вычисления выполняются локально, что снижает зависимость от качества сетевого соединения.

Реализация системы на портативных платформах требует решения технических задач, связанных с оптимизацией энергопотребления. Механизм адаптивного выбора конфигураций может учитывать критерий энергетической эффективности: система автоматически регулирует интенсивность вычислений в зависимости от уровня заряда батареи и режима работы устройства. При критически низком уровне заряда система переключается на альтернативные конфигурации ЭА, обеспечивающие приемлемое качество решений при минимальных энергозатратах.

Развитие вычислительных платформ и технологий периферийных вычислений создает предпосылки для внедрения АИС ЦУО в состав носимых устройств и встраиваемых систем Интернета вещей. Распределённая архитектура, при которой отдельные узлы выполняют локальную оптимизацию с последующей координацией через облегчённые протоколы обмена, представляет перспективное направление развития системы. Такой подход обеспечивает масштабируемость решений при сохранении адаптивности к специфике задач.

4.4 Выводы

В данной главе представлена апробация интеллектуальной системы для решения целочисленных задач оптимизации. В результате апробации были получены следующие результаты:

1. Внедрение системы в управление региональной сетью грузоперевозок с учетом более 20 критериев позволило снизить долю недопустимых решений более чем на 70% по сравнению с базовыми реализациями и повысить процент выполнения заказов в срок с 82% до 96%, что подтверждает

эффективность механизма верификации промежуточных решений с применением штрафных функций.

2. Сравнительный анализ с существующими программными пакетами оптимизации конечных автоматов показал, что АИС ЦУО обеспечивает наибольшее значение МКС при минимальной вычислительной скорости, что свидетельствует о преимуществах интеграции интеллектуальной модели выбора конфигураций и механизма верификации решений.

3. Проведена апробация разработанной адаптивной интеллектуальной системы целочисленной условной оптимизации на примерах управления конечными автоматами системы обеспечения телефонной связи. Результаты показали снижение прогнозируемого времени доставки на 18% и снижение серверной нагрузки на 6%. На модули программного обеспечения получены свидетельства о регистрации программы для ЭВМ.

4. Проведенная апробация показала статистически значимое различие между интеллектуальным и случайным выбором конфигураций эволюционных алгоритмов, при этом к завершению вычислительного процесса интеллектуальный выбор обеспечивает прирост эффективности примерно на 53% относительно случайного подхода.

5. При внесении модификаций в работу АИС ЦУО, ее применение перспективно в области теории игр: алгоритмы целочисленной оптимизации способно находить оптимальные стратегии для конкурентов с разными приоритетами, учитывая ограничения и балансируя между противоречивыми критериями;

6. В теории принятия решений эволюционные алгоритмы помогают находить компромиссные решения, минимизируя субъективность и учитывая вероятностные сценарии.

ЗАКЛЮЧЕНИЕ

В процессе выполнения диссертационного исследования получены следующие основные результаты:

1. Проведен системный анализ применимости эволюционных алгоритмов к задачам целочисленной условной оптимизации, выявлены их преимущества и недостатки, определены направления модификации, что обеспечило теоретическое обоснование выбора алгоритмов и стратегий их применения.

2. Созданы средства формализации представления ограничений, позволяющие описывать правила в виде предикатов произвольной степени вложенности и интеграцию штрафных функций в работу эволюционных алгоритмов, что обеспечивает корректную интерпретацию условий задачи с учетом контекста вычислений.

3. Разработана структура системы принятия решений в условиях многокритериальных ограничений, включая механизм применения штрафов по каждому критерию. Внедрение в систему управления цепочками поставок позволило снизить долю недопустимых решений более чем на 70% по сравнению с базовыми реализациями.

4. Модифицированы и адаптированы эволюционные алгоритмы (генетический, муравьиный, пчелиный и имитации отжига) для решения задач целочисленной условной оптимизации. В алгоритмы интегрирован механизм верификации промежуточных решений и реализован интерфейс взаимодействия с системой выбора и переключения алгоритмов.

5. Разработана интеллектуальная модель вариативного выбора конфигураций эволюционных алгоритмов на основе формализованного представления ограничений задачи с использованием трансформерной архитектуры, что превращает систему целочисленной условной оптимизации в адаптивный искусственный интеллект прикладного уровня.

6. Проведена апробация разработанной адаптивной интеллектуальной системы целочисленной условной оптимизации на примерах управления

логистикой и конечными автоматами системы обеспечения телефонной связи. Результаты показали снижение прогнозируемого времени доставки на 18% и снижение серверной нагрузки на 6%. На модули программного обеспечения получены свидетельства о регистрации программы для ЭВМ.

В соответствии с целью диссертационной работы было достигнуто повышение эффективности решения задач целочисленной условной оптимизации за счет разработки интеллектуальной системы.

СПИСОК ЛИТЕРАТУРЫ

1. Богданова, Е. Л. Оптимизация в проектном менеджменте: линейное программирование / Е. Л. Богданова, К. А. Соловейчик, К. Г. Аркина. – Санкт-Петербург : Университет ИТМО, 2017. – 124 с.

2. Ковалев, М. М. Дискретная оптимизация: целочисленное программирование / М. М. Ковалев. – Санкт-Петербург : ЛЕНАНД, 2023. – 192 с.

3. Nemhauser, G. L. Integer and combinatorial optimization / G. L. Nemhauser, L. A. Wolsey. – New York : Wiley, 1999. – 763 p.

4. Горовых, И. И. Гибридный подход к решению NP-сложных задач настройки производства / И. И. Горовых, С. Н. Горовых // Молодой исследователь Дона. – 2024. – Т. 9, № 1. – С. 10–19.

5. Fedyanin, D. N. Применение эволюционных алгоритмов в логистике : учебный курс / D. N. Fedyanin. – Москва : НИУ ВШЭ, 2023. – 156 с.

6. Potvin, J.-Y. The Traveling Salesman Problem and Neural Networks: An Efficient Implementation / J.-Y. Potvin, S. Bengio // Neural Computation. – 1996. – Vol. 8, № 5. – P. 987–1008.

7. Мицель, А. А. Эвристические методы оптимизации / А. А. Мицель. – Томск : ТУСУР, 2022. – 168 с.

8. Blum, C. Metaheuristics in combinatorial optimization: overview and conceptual comparison / C. Blum, A. Roli // ACM Computing Surveys. – 2003. – Vol. 35, № 3. – P. 268–308.

9. Мельников, Б. Ф. О классической версии метода ветвей и границ / Б. Ф. Мельников, Е. А. Мельникова // Компьютерные инструменты в образовании. – 2021. – № 2. – С. 5–28.

10. Хоролич, Г. Б. Эволюционные алгоритмы решения задач смешанной целочисленной оптимизации : автореферат диссертации на соискание ученой степени кандидата технических наук : 05.13.18 / Хоролич Геннадий Борисович. – Красноярск, 2002. – 18 с.

11. Eiben, A. E. Introduction to evolutionary computing / A. E. Eiben, J. E. Smith. – Berlin : Springer, 2003. – 299 p.

12. Jin, Y. A comprehensive survey of surrogate-assisted evolutionary computation / Y. Jin // Swarm and Evolutionary Computation. – 2011. – Vol. 1, № 1. – P. 61–71.

13. Звонков, В. Б. Генетический алгоритм с автоматической настройкой типа селекции и уровня мутации / В. Б. Звонков, Е. С. Семенкин // Актуальные проблемы авиации и космонавтики. – 2010. – Т. 1, № 6. – С. 245–246.

14. Шерстнев, П. А. Самоконфигурируемые алгоритмы генетического программирования / П. А. Шерстнев, Е. С. Семенкин // Сибирский аэрокосмический журнал. – 2025. – Т. 26, № 1. – С. 112–125.

15. Привалов, К. С. Гибридные методы оптимизации: адаптивное управление эволюционным процессом с использованием ИНС / К. С. Привалов // Инженерный вестник Дона. – 2025. – № 3. – С. 45–56.

16. Семенкин, Е. С. Коэволюционный алгоритм для задач условной и многокритериальной оптимизации / Е. С. Семенкин, Р. Б. Сергиенко // Программные продукты и системы. – 2010. – № 4. – С. 81–89.

17. Deb, K. Multi-objective optimization using evolutionary algorithms / K. Deb. – Chichester : Wiley, 2001. – 497 p.

18. Coello, C. A. C. Applications of multi-objective evolutionary algorithms / C. A. C. Coello, G. B. Lamont. – Singapore : World Scientific, 2004. – 516 p.

19. Васильев, Ф. П. Методы оптимизации / Ф. П. Васильев. – Москва : Факториал Пресс, 2002. – 824 с.

20. Хабарова, И. В. Исследование и разработка алгоритмов эволюционного моделирования с динамическими параметрами : автореферат диссертации на соискание ученой степени кандидата технических наук : 05.13.18 / Хабарова Ирина Владимировна. – Самара, 2022. – 20 с.

21. Вирсански Э. Генетические алгоритмы на Python / пер. с англ. А.А. Слинкина. – М. ДМК Пресс, 2020. – 286 с.: ил.

22. Li X. An Empirical Study on the Impact of Problem Size and Pareto Front Shape on the Performance of Multi-Objective Evolutionary Algorithms / X.

Li, Tang, K., Ochoa, G., Yao, X., Burke, E. K. // IEEE Transactions on Evolutionary Computation. – 2015. – №19(3). – С. 326 – 341.

23. Скакалина, Е. В. Использование генетических алгоритмов для решения задачи оптимизации транспортных перевозок / Е. В. Скакалина // Известия СПбГЭТУ ЛЭТИ. – 2015. – № 4. – С. 31 – 36.

24. Кажаров, А. А. Муравьиные алгоритмы для решения транспортных задач / А. А. Кажаров, В. М. Курейчик // Известия Российской академии наук. Теория и системы управления. – 2010. – № 1. – С. 32 – 45.

25. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. – 2003. – № 4 (4) – С. 70 – 75

26. Colorni A. Distributed Optimization by Ant Colonies, actes de la premiere conference euriopeenne sur la vie artificielle / A. Colorni, M. Dorigo, V. Maniezzo // Elsevier Publishing, Paris, France. - 1991. – С. 134-142.

27. Karaboga D.D. An Idea Based On Honey Bee Swarm for Numerical Optimization // Technical Report-TR64, Erciyes University, Engineering Faculity, Computer Engineering Department. - 2005

28. Савин А. Н. Применение алгоритма оптимизации методом имитации отжига на системах параллельных и распределённых вычислений / А. Н. Савин, Н. Е. Тимофеева // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. – 2012. – Т. 12, № 1. – С. 110-116

29. Лопатин, А. С. Метод отжига / А. С. Лопатин // Стохастическая оптимизация в информатике. – 2005. – Т. 1. – С. 133 – 149

30. Sabry A.H. A Perfomance Comparsion of GA and ACO Applied to TSP / A.H. Sabry, J. Benhra, H.E. Hassani // International Journal of Computer Applications. – 2015. - №117(May). – С. 28 – 35

31. Баранов, Д. А. Программная реализация задачи линейной оптимизации на примере муравьиного алгоритма / Д. А. Баранов, М. А. Белых, В. Ф. Барабанов // Оптимизация и моделирование в автоматизированных системах : труды Международной молодежной научной школы, Воронеж, 20–21 декабря

2023 года. – Воронеж: Воронежский государственный технический университет, 2024. – С. 19 – 22.

32. Baranov D.A. Software implementation of linear optimization problem using ant colony algorithm // Антропоцентрические науки в образовании: вызовы трансформации, ресурсы. – 2024 – С. 327 – 329

33. Курейчик В.М. Об алгоритмах решения задачи коммивояжёра в сети интернет / В.М. Курейчик, Ю.А. Логунова // Вестник РГРТУ. 2019. № 68 – С. 37 – 43

34. Программная реализация задачи линейной оптимизации на базе муравьиного алгоритма / Д. А. Баранов, М. А. Белых, В. Ф. Барабанов [и др.] // Вестник Воронежского государственного технического университета. – 2023. – Т. 19, № 6. – С. 53 – 58.

35. Павленко А.И., Титов Ю.П. Сравнительный анализ модифицированных методов муравьиных колоний // Прикладная информатика. – 2012. – №4 (40). – С. 100 – 112.

36. Оптимизация на примере. Муравьиный алгоритм (ACS) против Метода отжига. Часть 2. – URL: https://www.pvsm.ru/matlab/204887

37. Toaza B. A. review of metaheuristic algorithms for solving TSP-based scheduling optimization problems / A.B. Toaza, D. Esztergár-Kiss // Applied Soft Computing. – 2023. – № 148.

38. Белых, М. А. Решение задачи коммивояжера вариативным муравьиным алгоритмом / М. А. Белых, Д. А. Баранов // Информационные технологии моделирования и управления. – 2024. – Т. 136, № 2. – С. 116 – 119

39. Скаков Е.С., Малыш В.Н. Модифицированный алгоритм пчелиной колонии ABC для проектирования топологии беспроводной сети // Труды Международного симпозиума «Надежность и качество». – 2016. – Т. 1. – С. 293 – 296.

40. Karaboga D. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems // Lecture Notes in Computer Science. – 2007. - №4529 – С. 789 – 798

41. Баранов, Д. А. Модификация алгоритма пчелиной колонии для решения транспортной задачи оптимизации с динамическим количеством критериев / Д. А. Баранов // Нано-био-технологии. Тепло- и электроэнергетика. Математическое моделирование : Сборник статей III международной научно-практической конференции, Липецк, 27–28 февраля 2025 года. – Липецк: Липецкий государственный технический университет, 2025. – С. 286-294.

42. Ходашинский И.А. Алгоритмы муравьиной и пчелиной колонии для обучения нечетких систем / И.А. Ходашинский, И.В. Горбунов, П.А. Дудин // Доклады Томского государственного университета систем управления и радиоэлектроники. – 2009. - № 2 (20). – С. 157 – 161

43. Adewole A.P. A Comparative Study of Simulated Annealing and Genetic Algorithm for solving the Travelling Salesman Problem / Adewole A.P., Kehinde O., T.O. Egunjobi, K.M. Ng // International Journal of Applied Information System. – 2012. – С. 6 – 12

44. Баранов, Д. А. Модификация алгоритма имитации отжига для решения многокритериальной транспортной задачи / Д. А. Баранов // Цифровые системы и модели: теория и практика проектирования, разработки и использования : Материалы международной научно-практической конференции, Казань, 10–11 апреля 2025 года. – Казань: Казанский государственный энергетический университет, 2025. – С. 757 – 761.

45. Vidal M. A domain-specific language for verifying software requirement constraints / M. Vidal, T. Massoni, F. Ramalho // Science of Computer Programming. – 2020. – Vol. 197. – 26 с.

46. Черч А. Введение в математическую логику. Том 1 / пер. с англ. – М.: Издательство иностранной литературы, 1960. – 485 с.

47. Łukasiewicz J. Philosophische Bemerkungen zu mehrwertigen Systemen des Aussagenkalküls // Comptes Rendus des Séances de la Société des Sciences et des Lettres de Varsovie. Cl. III. — 1930. — Vol. 23. — pp. 51 – 77.

48. Баранов, Д. А. Разработка языка описания ограничений, применяемых к многокритериальным задачам оптимизации / Д. А. Баранов //

Информационные технологии в строительных, социальных и экономических системах. – 2025. – № 3(37). – С. 137-142. – EDN WJRFBK.

49. Семенкин, Е. С. Программный комплекс адаптивных эволюционных алгоритмов моделирования и оптимизации сложных систем / Е. С. Семенкин, М. Е. Семенкина // Программные продукты и системы. – 2012. – № 4. – С. 73–77.

50. Яшин, С. Н. Метаэвристические алгоритмы в управлении инновациями : монография / С. Н. Яшин, Н. И. Яшина, Е. В. Кошелев, А. А. Иванов. – Нижний Новгород : ННГУ им. Н. И. Лобачевского, 2023. – 200 с.

51. Цыликин, А. С. Алгоритмы эволюционной оптимизации / А. С. Цыликин. – Москва : БХВ-Петербург, 2020. – 940 с.

52. Михайлов, А. Н. Эволюционные алгоритмы: эффективные подходы к решению задач оптимизации и управления в сложных системах / А. Н. Михайлов // Вестник науки. – 2020. – № 12 (81). – С. 727–735.

53. Гончаренко, В. А. Генетический алгоритм выбора оптимальной конфигурации RAID-массивов системы хранения данных предприятия / В. А. Гончаренко, А. Д. Хомоненко, Р. Абу Хасан // Учёные записки КнАГТУ. – 2025. – № I (81). – С. 50–60.

54. Дивеев, А. И. Эволюционные алгоритмы для решения задачи оптимального управления / А. И. Дивеев, С. В. Константинов // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. – 2017. – Т. 18, № 2. – С. 254–265.

55. Vaswani, A. Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar [et al.] // Advances in Neural Information Processing Systems. – 2017. – Vol. 30. – P. 5998–6008.

56. Белых, М. А. Разработка интеллектуальной системы оптимизации на основе эволюционных алгоритмов / М. А. Белых, Д. А. Баранов // Нано-биотехнологии. Теплоэнергетика. Математическое моделирование : Сборник статей международной научно-практической конференции, Липецк, 27–28

февраля 2024 года. – Липецк: Липецкий государственный технический университет, 2024. – С. 158-164

57. C++ быстрее и безопаснее Rust, Yandex сделала замеры // Хабр. – URL: https://habr.com/ru/articles/492410/

58. Rust vs. C++ на алгоритмических задачах / Хабр. URL: https://habr.com/ru/articles/344282/

59. Rust Vs. C++: Making the Best Choice for Your Projects. URL: https://www.koombea.com/blog/rust-vs-c/

60. Лучшие языки программирования для изучения в 2023 году | Голосование / Хабр. - URL: https://habr.com/ru/articles/722586/

61. TIOBE Index – TIOBE. URL: https://www.tiobe.com/tiobe-index/python/

62. pandas - Python Data Analysis Library. – URL: https://pandas.pydata.org/about/

63. Пасхавер Б. Pandas в действии. – СПб.: Питер, 2023. – 512 с.

64. Манцнер Т. Визуализация данных: Полный и исчерпывающий курс для начинающих // Эксмо. – 2023. – 464 с.

65. Мюллер Д.П. Python и наука о данных для чайников / Д.П. Мюллер, Л. Массарон // Диалектика. – 2020. – 512 с.

66. Mission Statement — Matplotlib 3.10.1 documentation. – URL: https://matplotlib.org/stable/project/mission.html

67. An introduction to seaborn — seaborn 0.13.2 documentation. - URL: https://seaborn.pydata.org/tutorial/introduction.html

68. Жерон О. Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. 2-е издание // Диалектика. – 2020. – 1040 с.

69. Ракша С. Машинное обучение с PyTorch и Scikit-Learn / С. Ракша, Ю. Лю, В. Мирджалили // Фолиант. – 2024. – 688 с.

70. Constrained Optimization Algorithms in Rust : r/rust. - URL: https://www.reddit.com/r/rust/comments/px3jcl/constrained_optimization_algorithms_in_rust/?rdt=44412

71. rand - Rust. - URL: https://docs.rs/rand/latest/rand/

72. random_choice - Rust. - URL: https://docs.rs/random_choice/latest/random_choice

73. JSON Introduction. - URL: https://www.w3schools.com/js/js_json_intro.asp

74. Ахтырский, А. А. Создание и обработка файлов в формате JSON / А. А. Ахтырский, Т. В. Волобуева // Информационные технологии в строительных, социальных и экономических системах. – 2024. – № 1-2(31-32). – С. 128 – 133

75. Overview · Serde. - URL: https://serde.rs/

76. serde_json - Rust. URL: https://docs.rs/serde_json/latest/serde_json/

77. Aho A.V. Compilers: Principles, Techniques, and Tools, 2nd Edition / A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman // Pearson. – 2011 – 1009 С.

78. Макнамара Т. Rust в действии: Пер. с англ. – СПб.: БХВ-Петербург, 2023. — 528 с.: ил.

79. Peters T. Timsort: A new approach to merging sorted runs. // Python Software Foundation, 2002.

80. Седжвик Р., Уэйн К. Алгоритмы на Java. Том 1. Основы. 4-е издание. М.: Вильямс, 2013. 720 с.

81. Кормен Т. Х., Лейзерсон Ч. Э., Ривест Р. Л., Штайн К. Алгоритмы: построение и анализ. 3-е издание. М.: Вильямс, 2013. 1328 с.

82. Белых, М.А. Сравнительный анализ работы эволюционных алгоритмов при решении многокритериальной транспортной задачи без ограничений / М. А. Белых, Д.А. Баранов, В. Ф. Барабанов // Вестник Воронежского государственного технического университета. – 2024. – Т. 20, № 4. – С. 43 – 48. – DOI 10.36622/1729-6501.2024.20.4.006.

83. ryanjoneil/tsplib. - URL: https://github.com/ryanjoneil/tsplib

84. Белых, М А. Сравнительный анализ эволюционных алгоритмов при решении многокритериальной транспортной задачи с временными ограничениями / М. А. Белых, Д. А. Баранов, В. Ф. Барабанов // Системы управления и информационные технологии. – 2024. – № 4(98). – С. 61 – 66.

85. Баранов Д.А. Сравнительный анализ методов эволюционного проектирования в программном обеспечении для решения многокритериальных задач оптимизации // Моделирование, оптимизация и информационные технологии. – 2025. – Т. 13, № 2(49). – DOI: 10.26102/2310-6018/2025.49.2.008

86. Сальникова, К. В. Анализа массива данных с помощью инструмента визуализации "Ящик с усами" / К. В. Сальникова // Universum: экономика и юриспруденция. – 2021. – № 6(81). – С. 11 – 17. – DOI 10.32743/UniLaw.2021.81.6.11778.

87. Баранов, Д. А. Исследование эффективности эволюционных алгоритмов в задачах дискретной оптимизации высокой размерности / Д. А. Баранов // Моделирование, оптимизация и информационные технологии. – 2025. – Т. 13, № 3(50). – DOI 10.26102/2310-6018/2025.50.3.048. – EDN UGYRCS.

88. Баранов Д.А. Интеллектуальная система управления эволюционными алгоритмами в дискретных задачах оптимизации / Д.А. Баранов, Барабанов В.Ф. // Вестник Воронежского государственного технического университета. – 2025. – Т. 21, №4(102). – С. 39 – 44

89. Царев, Ф. Н. Метод построения управляющих конечных автоматов на основе тестовых примеров с помощью генетического программирования / Ф. Н. Царев // Информационно-управляющие системы. – 2010. – № 5(48). – С. 31 – 36. – EDN MVKNHV.

90. nkg447/DFA-Minimizer: Java program to Minimize deterministic finite automata, Convert NFA to DFA and graphically represent the results.. - URL: https://github.com/nkg447/DFA-Minimizer

91. rushilchoksi/DFA-Minimization: Python based utility to perform minimization of deterministic finite automata using equivalence theorem. - URL: https://github.com/rushilchoksi/DFA-Minimization

92. shysaur/fsmmin: Toolkit for Finite State Machines. - URL: https://github.com/shysaur/fsmmin

93. swaroopkml96/Finite-State-Machine-Minimization: Python program to visualize, run and minimize finite state machines. - URL: https://github.com/swaroopkml96/Finite-State-Machine-Minimization

94. Вартанов С.А., Ивин Е.А. Прикладная теория экономистов. – Вологда: ВолНЦ РАН, 2020. – 283 с.

95. Применение математической теории игр в системе поддержки принятия решений руководителем тушения пожара / И. М. Тетерин, Н. Г. Топольский, В. М. Климовцов, Ю. В. Прус // Технологии техносферной безопасности. – 2008. – № 6(22). – С. 9. – EDN MSMBZV.

96. Шабанов, А. Д. Использование теории игр в практике принятия управленческих решений / А. Д. Шабанов // Аллея науки. – 2020. – Т. 1, № 5(44). – С. 958-964. – EDN KHULQN.

97. Теории игр всегда будет, что вам предложить в качестве идей» – Новости – Международный институт экономики и финансов – Национальный исследовательский университет «Высшая школа экономики». - URL: https://icef.hse.ru/news/835735240.html

98. Принятие решений в условиях неопределенности - Home page of Lev V. Utkin. - URL: https://levutkin.github.io/teaching/decision-making

99. Уткин Л.В. Анализ риска и принятие решений при неполной информации. – СПб. Наука, 2007. – 404 с.

100. Ногин В.Д. Принятие решений при многих критериях. Учебно-методическое пособие. – СПб, Издательство «ЮТАС», 2007. – 104 с.

101. Халин В.Г. Теория принятия решений в 2 Т. Том 2: учебник и практикум для вузов. – Москва: Издательство Юрайт, 2020. – 431 с.

102. Как принимать решения в бизнесе: практические советы и кейсы. - URL: https://priceva.ru/blog/article/kak-izbezhat-oshibok-pri-prinyatii-reshenij-v-biznese-prakticheskie-sovety-i-kejsy

ПРИЛОЖЕНИЕ А Свидетельства о государственной регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025619236

Модуль верификации решения поставленной задачи на основе установленных ограничений

Правообладатель: ***Баранов Дмитрий Алексеевич (RU)***

Автор(ы): ***Баранов Дмитрий Алексеевич (RU)***

Заявка № **2025617972**
Дата поступления **10 апреля 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***14 апреля 2025 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f240f670bca2026
Владелец Зубов Юрий Сергеевич
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025619704

Модуль описания и обработки ограничений к многокритериальным задачам оптимизации

Правообладатель: ***Баранов Дмитрий Алексеевич (RU)***

Автор(ы): ***Баранов Дмитрий Алексеевич (RU)***

Заявка № **2025618397**
Дата поступления **10 апреля 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***17 апреля 2025 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f240f670bca2026
Владелец Зубов Юрий Сергеевич
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2025660546

Модель верификации решения многокритериальной задачи на основе установленных ограничений

Правообладатель: ***Баранов Дмитрий Алексеевич (RU)***

Автор(ы): ***Баранов Дмитрий Алексеевич (RU)***

Заявка № **2025618498**
Дата поступления **16 апреля 2025 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***24 апреля 2025 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0692e7c1a6300bf54f240f670bcca2026
Владелец **Зубов Юрий Сергеевич**
Действителен с 10.07.2024 по 03.10.2025

Ю.С. Зубов

ПРИЛОЖЕНИЕ Б Акты о внедрении результатов диссертационного исследования

ООО "Бренд 42"
Юридический адрес: 420107, г. Казань, ул. Островского, д.102, пом. 306
ИНН 1655373485
ОГРН 1161690159421
Телефон +79039605731 e-mail: anton.zimin@brand42.ru

Утверждаю
директор Зимин А.А.

АКТ

об использовании результатов диссертационной работы
Баранова Дмитрия Алексеевича
на соискание ученой степени кандидата технических наук

Результаты диссертационной работы Баранова Д.А, "Математическое и программное обеспечение вычислительных комплексов для эволюционного программирования и интеллектуальных систем", в том числе, предложенные автором:

модель активной нейросетевой архитектуры, интегрирующей гибридные эволюционные алгоритмы для ускоренного решения задач многокритериальной оптимизации. Модель сочетает представленные вариации параметров алгоритмов, что позволяет сократить время вычислений на 15-20% по сравнению с традиционными методами

алгоритм динамического обучения (дообучения) нейросети), обеспечивающий адаптацию к нестационарным производственным условиям и самоадаптацию гиперпараметров

метод учета многоуровневых ограничений в многокритериальных задачах доставки, включая:

жесткие ограничений (временные окна, приоритетность);

мягкие ограничения (стоимость логистики, экологические нормы);

динамические ограничения (изменение маршрутов из-за погодных условий). Для их обработки применен комбинированный подход с использованием штрафных функций;

используются в разработках систем управления цепями поставок и интеллектуальных транспортно-логистических комплексов. Внедрение позволило повысить точность прогнозирования времени доставки на 12-18% за счет анализа исторических данных и симуляции сценариев, снизить влияние человеческого фактора в различных ситуациях, а также оптимизировать маршруты с учетом 20+ параметров (загруженность, приоритет, топливная эффективность и др.)

Директор _____ Зимин А.А.

САТЕЛПРО

129090, Москва, ул. Щепкина, дом 28, пом.I, 2 этаж, комн. №4 и 5
Тел: +7 (495) 788-18-91, Факс: +7 (495) 788-18-91
E-mail: hdlo@ecosoft.pro, www.ecosoft.pro

Справка о внедрении

Настоящим подтверждаем, что результаты диссертационного исследования **Баранова Дмитрия Алексеевича**, связанные с многокритериальной оптимизацией, а именно:

- Методы эволюционного проектирования для решения многокритериальных задач;
- Язык описания ограничений к постановке задачи оптимизации с возможностью задания логических выражений любого уровня вложенности;
- Модель верификации решения поставленной задачи на основе установленных ограничений;

внедрены в практическую деятельность ООО «Сател ПрО» для оптимизации маршрутов передачи голосовых и цифровых данных с учетом многокритериальных ограничений (задержка, нагрузка каналов и пр.). Это помогло снизить требуемую вычислительную мощность серверов для программного комплекса РТУ на 6%.

Генеральный директор

ООО «САТЕЛ ПрО» Романов Р. Ю.

Общество с ограниченной ответственностью «Стартап»
ИНН 1656117815 ОГРН 1211600013250
420107, РЕСПУБЛИКА ТАТАРСТАН, Г. КАЗАНЬ, УЛ. ОСТРОВСКОГО, Д. 102, ПОМЕЩ. 406

СПРАВКА О ВНЕДРЕНИИ

Настоящим подтверждаем, что результаты диссертационного исследования Баранова Дмитрия Алексеевича "математическое и программное обеспечение эволюционных алгоритмов с использованием интеллектуальных технологий", а именно:

- интеллектуальная модель подбора и переключения эволюционных алгоритмов для решения многокритериальных задач оптимизации конечных автоматов
- алгоритм нормализации значений в нескольких степенях свободы для учета множества критерий задачи

внедрены в практическую деятельность ООО «Стартап»

Генеральный директор Миленький К.К.

УТВЕРЖДАЮ
Проректор по учебной работе
ФГБОУ ВО «ВГТУ»
_____ А.И. Колосов
«10» 11 2025 г.

А К Т
внедрения результатов кандидатской диссертации в учебный процесс ФГБОУ ВО «Воронежский государственный технический университет»

Тема диссертации: «Интеллектуализация системы целочисленной линейной оптимизации на основе эволюционных алгоритмов»

Автор: Баранов Дмитрий Алексеевич

Научный руководитель: Барабанов Владимир Федорович

Выполненной в ФГБОУ ВО «Воронежский государственный технический университет» на кафедре автоматизированных и вычислительных систем в рамках основного научного направления «Информатика и вычислительная техника»

В период с «_11_» __марта__ 2025 г. по н.в. внедрены в учебный процесс кафедры по группе научных специальностей 2.3 «Информационные технологии и телекоммуникации», научной специальности 2.3.1 «Системный анализ, управление и обработка информации, статистика», на основании решения кафедры АВС от «_21_» __октября__ 2025 г., протокол №_4_.

1. Вид результатов, внедренных в учебный процесс: совокупность знаний и представлений по теме диссертационного исследования.

2. Область применения: лабораторный практикум и лекционный курс по дисциплинам «Проектная деятельность» и «Интеллектуальные системы», выполнение курсовых проектов, выпускных квалификационных работ.

3. Форма внедрения: разработанные в диссертационном исследовании алгоритмические программные модули и правила формализации ограничений были внедрены в образовательный процесс в виде лабораторных и практических занятий на курсах, посвященных методам оптимизации и интеллектуальным системам. В частности, материалы использовались при обучении студентов решению многокритериальных

задач оптимизации с использованием эволюционных алгоритмов, а также адаптации решений под пользовательские ограничения. Кроме того, разработан учебный программный комплекс, демонстрирующий работу метода имитации отжига, генетического, муравьиного и пчелиного алгоритмов и способ их интеллектуального переключения.

4. Эффект от внедрения. Повышение качества образования: использование разработанного программного комплекса позволило студентам наглядно изучать методы эволюционного проектирования и их применение для решения многокритериальных задач с ограничениями. Визуализация процесса решения задачи и возможность гибкого задания параметров и ограничений способствовали лучшему пониманию процессов оптимизации и повысили интерес к дисциплине. Также внедрение языка описания ограничений способствовало развитию навыков формализации задач и системного подхода в проектировании интеллектуальных систем.

Научный руководитель диссертанта

_____ Барабанов В.Ф.
(подпись, Ф.И.О.)

«9» 11 2025 г.

Диссертант

_____ Баранов Д.А.
(подпись, Ф.И.О.)

«9» 11 2025 г.

Начальник УМУ

_____ Скляров К.А.
(подпись, Ф.И.О.)

«10» 11 2025 г.

Декан ФИТКБ

_____ Бредихин А.В.
(подпись, Ф.И.О.)

«11» 11 2025 г.

Заведующий кафедрой АВС

_____ Барабанов В.Ф.
(подпись, Ф.И.О.)

«9» 11 2025 г.