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

На правах рукописи

Матренин Павел Викторович

# РАЗРАБОТКА АДАПТИВНЫХ АЛГОРИТМОВ РОЕВОГО ИНТЕЛЛЕКТА В ПРОЕКТИРОВАНИИ И УПРАВЛЕНИИ ТЕХНИЧЕСКИМИ СИСТЕМАМИ

05.13.01 – Системный анализ, управление и обработка информации
(в отраслях информатики, вычислительной техники и автоматизации)

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель
доктор технических наук, профессор
Манусов Вадим Зиновьевич

Новосибирск – 2018

## Оглавление

# ВВЕДЕНИЕ

**Актуальность темы.** Развитие технических систем приводит к усложнению задач управления и повышению требований к эффективности их решения. Поэтому необходимо совершенствовать средства решения задач оптимизации, управления и обработки информации. При разработке крупных транспортных, производственных, информационных, энергетических и других сложных технических систем возникают задачи, связанные с оптимизацией взаимодействия объектов и их функционированием, что приводит к необходимости решать оптимизационные задачи с использованием эффективных математических методов оптимизации и современных средств вычислительной техники.

Решение оптимизационных задач проектирования и управления техническими системами необходимо для повышения технических и экономических показателей эффективности систем от отдельных устройств до глобальных систем, например, энергетической системы всей страны. Электроэнергетические системы выделяются особой сложностью среди прочих технических систем. Актуальность повышения энергоэффективности признана на государственном уровне законом РФ № 261 – ФЗ «Об энергосбережении и о повышении энергетической эффективности, и о внесении изменений в отдельные законодательные акты Российской Федерации» от 23.10.2009 г. В области электроэнергетики существует множество задач оптимизации различных классов, причем из-за высокой сложности систем электроснабжения эти задачи часто являются нелинейными, недифференцируемыми, многокритериальными, многофакторными. Трудоемкость таких задач, как правило, экспоненциально увеличивается с увеличением числа элементов системы, кроме того, многие задачи необходимо решать в режиме реального времени. Наиболее важными оптимизационными задачами в электроэнергетике являются: задача выбора конфигурации электросетей, распределение нагрузки между источниками электроэнергии, определение оптимального размещения элементов систем, оптимальная компенсация реактивной мощности, задача определения точек

размыкания сети, управление коэффициентами трансформаторов. Вопросам оптимального проектирования и оптимизации режимов работы электроэнергетических систем посвящены работы Д.А. Арзамасцева, А.С. Бердина, А.А. Герасименко, В.М. Манусова, С.А. Решетова, Е.В. Цветкова, В.С. Хачатряна, В.А. Стенникова. Большая часть работ основана на применении детерминированных методов, таких как множители Лагранжа, разбиение задачи на подзадачи, метод ветвей и границ, градиентные методы. Из-за наличия неопределенностей в рассматриваемых системах часто применяется нечеткая логика. Высокая вычислительная сложность, неопределенности и сложные топологии пространств поиска решений ограничивают применение указанных методов. Ограничения связаны как с недостаточной эффективностью или скоростью работы, так и с высокой трудоемкостью применения методов.

Эвристические методы, основанные на некотором экспертном знании, позволяют быстро находить приближенные решения, но, как показано в работах В.Л. Береснева, Ф. Гловера, Д. Джелатта, Ю.А. Кочетова, Л.А. Растригина, С. Скиены, использование простых эвристических методов и жадных эвристик может приводить как к глобально оптимальным, так и неудовлетворительным решениями, поэтому необходимо применять более эффективные методы оптимизации. В многих областях для получения близкого к оптимальному решения за время, допустимое для функционирования системы, успешно применяются стохастические эвристические методы оптимизации, такие как алгоритм имитации отжига, эволюционные и роевые алгоритмы. Они относятся к методам локального поиска и основаны на идеях, взятых из природы. При этом их эффективность и асимптотическая сходимость к глобальному оптимуму задачи доказывается с применением конечных цепей Маркова. Методы обладают стохастической природой, что делает их применение нетривиальной задачей, так как для каждого алгоритма и различных его реализаций и для каждого класса оптимизационных задач скорость работы, точность, сходимость, влияние условий задачи и эвристических коэффициентов требуют особого исследования.

Термин «роевой интеллект» предложен в конце 1980-х годов Х. Бени и Ц. Вангом. Первыми роевыми алгоритмами, получившими распространение, стали алгоритм роя частиц Дж. Кеннеди и Р. Эберхарта и алгоритм колонии муравьев М. Дориго. В последние годы возник ряд новых роевых алгоритмов, но еще не созданы четкие терминология и классификация. По той же причине существует множество нерешенных задач для исследования. Наибольшей сложностью в применении роевых алгоритмов является их настройка и доработка для различных видов оптимизационных задач, подбор значений коэффициентов алгоритмов для получения высокой эффективности на различных классах задач, как показано во многих исследованиях, среди которых можно выделить работы Л.А. Гладкова, В.В. и В.М. Курейчиков, А.П. Карпенко, В.Б. Лебедева, М. Дориго, М. Педерсена, Ю. Ши, Р. Эберхарта. Для различных задач оптимизации лучше подходят различные роевые алгоритмы, но методик их выбора не существует. В результате указанных сложностей, которые можно обобщенно назвать сложностями адаптации, алгоритмы роевого интеллекта недостаточно эффективно применяются на практике. Поэтому актуальность работы заключается в повышении адаптивных свойств алгоритмов роевого интеллекта и создании инструмента для решения задач оптимизации.

Решению задач оптимизации технических систем с помощью алгоритмов роевого интеллекта посвящены работы многих авторов: Л.А. Гладкова, А.А. Кажарова, Ю.А. Кочетова, Е.А. Кочегуровой, В.В. Курейчика, В.М. Курейчика, Б.К., В.Б. и О.Б. Лебедевых, О.Г. и Э.А. Монаховых, В.Г. Секаева, В.Д. Фроловского, И.А. Ходашинского, С.Д. Штовбы, A. Abraham, A. Carlisle, C. Chen, M. Dorigo, G. Dozier, R.C. Eberhart, J. Kennedy, Y. Lee, M. Pedersen, Y. Valle, Y. Shi, X.-S. Yang, H. Yoshida и других.

**Объект исследования** – проектирование и управление электротехническими системами.

**Предмет исследования.** Предметом исследования являются задачи оптимизации электроэнергетических систем и применение алгоритмов роевого интеллекта для их решения.

**Основная идея работы.** В диссертационной работе на основе системного анализа предложен новый подход к описанию, реализации и практическому применению алгоритмов роевого интеллекта, основанный на разработке унифицированной модели роевых алгоритмов, интерфейса между алгоритмами и задачами и эволюционной адаптации алгоритмов для решения практически ориентированных оптимизационных задач проектирования и управления техническими системами на примере электроэнергетических систем.

**Цель и задачи исследования.** Целью работы является улучшение качества функционирования технических систем за счет разработки и применения адаптивных алгоритмов роевого интеллекта в решении оптимизационных задач. Поставленные в работе задачи:

1) исследовать существующие методы оптимизации и особенности оптимизационных задач в области проектирования и управления техническими системами;

2) изучить особенности алгоритмов роевого интеллекта в решении оптимизационных задач, отличающихся сложными взаимосвязями, высокой размерностью и вычислительной сложностью моделей, наличием нескольких критериев, динамическими и стохастическими свойствами;

3) выполнить системный анализ концепции роевого интеллекта, провести четкую классификацию алгоритмов, ввести терминологию, выделить общие и частные особенности роевых алгоритмов;

4) модифицировать алгоритмы роевого интеллекта для их простого, быстрого и эффективного применения на практике, разработать метод повышения их эффективности за счет адаптации под условия решаемых задач;

5) апробировать адаптивные алгоритмы роевого интеллекта в решении задач проектирования и управления в области электроэнергетики для повышения технико-экономических показателей электроэнергетических систем;

6) подтвердить эффективность эволюционной адаптации алгоритмов роевого интеллекта на NP-трудных тестовых задачах оптимизации, провести сравнение с результатами других авторов;

7) создать программные реализации разработанных алгоритмов в формате универсальной интегрируемой библиотеки для применения в решении задач оптимизации и разработать приложения, демонстрирующие применение алгоритмов роевого интеллекта в электроэнергетике, календарном планировании, а также приложения для визуализации работы роевых и алгоритмов для их исследования и применения в учебном процессе.

**Методы исследования.** Для решения поставленных задач использовались как общенаучные (анализ, синтез, абстрагирование), так и специальные методы. К последним относятся системный анализ, математическое и компьютерное моделирование, вычислительный эксперимент, мета-оптимизация. Для решения оптимизационных задач применены роевые и эволюционные методы.

**Научная новизна результатов, изложенных в диссертации:**

1) впервые разработана общая математическая модель описания, программной реализации и практического применения алгоритмов роевого интеллекта с использованием методов системного анализа, а также единая для роевых алгоритмов терминология и система условных обозначения, что позволяет провести классификацию роевых алгоритмов, выделять общие отличительные особенности роевых алгоритмов;

2) показано, что эффективная адаптация алгоритмов роевого интеллекта к условиям решаемых задач оптимизации технических систем требует устранения зависимостей между роевым алгоритмом и моделью оптимизируемой системы, для этого предложен новый интерфейс между алгоритмом и моделью и схема их взаимодействия;

3) показана целесообразность мета-оптимизации алгоритмов роевого интеллекта на основе оригинального метода эволюционной адаптации значений их эвристических параметров под отдельные виды решаемых задач в области проектирования и управления электроэнергетическими системами; показана устойчивость предложенного способа мета-оптимизации.

**Теоретическая и практическая значимость работы**. Проведена систематизация алгоритмов роевого интеллекта, созданы математическая модель и

программные реализации алгоритмов роевого интеллекта для решения оптимизационных задач в различных областях. Разработан унифицированный интерфейс между программными реализациями алгоритмов роевого интеллекта и моделями оптимизируемых объектов для различных языков программирования и программных комплексов, а также изложены подходы, позволяющие повысить эффективность и быстродействие роевых алгоритмов при их программной реализации. Показана эффективность применения адаптивных алгоритмов роевого интеллекта в задачах оптимизации электроэнергетическим систем: в оптимизации глубокой компенсации реактивной мощности и управления коэффициентами трансформации. Практические результаты могут быть использованы в проектных организациях, в системах электроснабжения предприятий для решения оптимизационных задач проектирования и управления.

**Достоверность и обоснованность полученных результатов** подтверждаются корректностью применяемых методик исследования, использованием оптимизационных алгоритмов, эффективность которых математически и экспериментально доказана, корректным использованием математического и компьютерного моделирования с применением современных программных средств, проведенными вычислительными экспериментами, дающими воспроизводимые результаты, и анализом численных результатов. Эффективность и достоверность разработанных алгоритмов и программных приложений подтверждается их применением на практике.

**Реализация и внедрение результатов работы.** Результаты работы применены для проектирования оптимального размещения источников реактивной мощности в системе электроснабжения подстанций АО «Уральский электрохимический комбинат». Полученные результаты рекомендованы для рассмотрения при проведении работ по техническому перевооружению подстанций комбината, согласно проведенному исследованию, это позволит снизить потери активной мощности в линиях электропередач на 15–22 % со сроком окупаемости 3 года для нерегулируемых установок компенсации реактивной мощности и 3,5 года для регулируемых установок.

Созданные приложения визуализации алгоритмов локального поиска и учебное пособие «Методы стохастической оптимизации» внедрены в учебный процесс в Новосибирском государственном техническом университете на кафедре «Системы электроснабжения предприятий» при преподавании дисциплин «Интеллектуальные системы электроснабжения», «Оптимизация систем электроснабжения» и «Системный анализ в электроэнергетике», а также при написании бакалаврских, магистерских диссертаций и в научных исследованиях аспирантов кафедры.

**Положения, выносимые на защиту:**

1) разработанная обобщенная схема описания алгоритмов роевого интеллекта, их функционирования и взаимодействия с оптимизируемыми объектами позволили провести классификацию популяционных алгоритмов, упростили реализацию и применение алгоритмов роевого интеллекта;

2) эффективная адаптация алгоритмов роевого интеллекта имеет два основных аспекта: во-первых, устранение тесных зависимостей между реализацией алгоритмов и моделями оптимизируемых систем, во-вторых, настройка значений эвристических коэффициентов роевых алгоритмов под определенный класс задач – первое достигается путем определения специального интерфейса между алгоритмом и задачей, так что задача оптимизации становится для алгоритма черным ящиком, как и алгоритм для задачи, второе за счет мета-оптимизации эвристических коэффициентов;

3) эффективным методом мета-оптимизации алгоритмов роевого интеллекта является генетический алгоритм, позволяющий выполнять адаптацию алгоритма роевого интеллекта под определенный класс оптимизационных задач;

4) разработанные адаптивные алгоритмы роевого интеллекта позволяют эффективно решать разнообразные оптимизационные задачи в электроэнергетике без трудозатрат на модификацию алгоритмов под отдельные классы задач, за счет автоматической адаптации алгоритмов;

5) разработан программный комплекс решения задач оптимизации, позволяющий повышать технико-экономические показатели функционирования

электроэнергетических систем, комплекс может быть интегрирован в различные системы моделирования, такие как «Simulink» и «RastrWin».

**Апробация результатов.** Работа докладывалась и обсуждалась на расширенных заседаниях кафедры «Системы электроснабжения предприятий» ФГБОУ ВО «Новосибирский государственный технический университет». Основные результаты диссертационного исследования были представлены на всероссийских и международных конференциях: Всероссийская научная школа-конференция «Состояние и пути развития российской энергетики» (Томск, 21–23 октября 2014 г.), XI Международная IEEE Сибирская конференция по управлению и связи SIBCON-2015 (Омск, 21–23 мая 2015 г.), VII Международная конференция «Электротехника. Электротехнология. Энергетика» (Новосибирск, 09–12 июня 2015 г.), X Международная научно-практическая конференция «Объектные системы» (Ростов-на-Дону, 10–12 мая 2015 г.), XIII международная научно-техническая конференция «Актуальные проблемы электронного приборостроения» (Новосибирск, 03–06 октября 2016 г.), 2nd International Conference on Energy Production and Management (Анкона, 06–08 сентября 2016 г), 2nd International Conference on Industrial Engineering, Applications and Manufacturing (Челябинск, 19–20 мая 2016 г.), 18th International Scientific Conference on Electric Power Engineering, (Коуты над Десной, 17–19 мая 2017 г.), II Международная конференция «Устойчивое развитие городов» (Екатеринбург, 19 мая 2017 г.).

**Публикации**. По материалам диссертации опубликовано 33 работы, из них 14 статей в журналах, включенных в Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук (в том числе 5 статей в зарубежных научных журналах, индексируемых Web of Science / Scopus), 4 свидетельства о государственной регистрации программы для ЭВМ, 6 статей в научных журналах, 2 статьи в сборниках научных трудов, 7 публикаций в сборниках материалов международных и всероссийских научных и научно практических конференций (в том числе 4 статьи в сборниках,

индексируемых Web of Science). Общий объем публикаций автора– 13,96 а.л., личный вклад автора – 10,28 а.л.

**Личное участие автора в получении результатов, указанных в диссертации.** Автор лично участвовал в построении математических моделей исследуемых алгоритмов роевого интеллекта, разработке алгоритма эволюционной адаптации роевых алгоритмов, их программной реализации, разработке прикладных приложений для проведения вычислительных экспериментов, решения прикладных задач оптимизации, визуализации алгоритмов, а также в подготовке публикаций и докладов по теме исследования. Обработка и интерпретация полученных теоретических и экспериментальных результатов выполнена лично автором.

**Соответствие диссертации паспорту научной специальности.** Диссертационное исследование выполнено в соответствии с паспортом специальности 05.13.01 «Системный анализ, управление и обработка информации» (отрасль наук: технические науки) и включает в себя оригинальные результаты из следующих областей исследований (номера соответствуют пунктам паспорта специальности):

4) «Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации»;

5) «Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений и обработки информации»;

10) «Методы и алгоритмы интеллектуальной поддержки при принятии управленческих решений в технических системах».

**Структура и объем работы.** Диссертация состоит из введения, пяти разделов, заключения, списка литературы из 182 наименований и четырех приложений. Общий объем работы составляет 197 страниц.

# 1. Обзор и анализ методов решения оптимизационных задач

В первом разделе рассмотрено современное состояние проблемы решения задач оптимизации. В подразделе 1.1 дан краткий исторический обзор развития методов оптимизации и приведены способы классификации методов. В подразделе 1.2 приведен обзор наиболее важных методов оптимизации от детерминированных методов, основанных на переборе решений, до стохастических эволюционных и роевых алгоритмов. В подразделе 1.3 описана история развития алгоритмов роевого интеллекта, приведена хронология возникновения различных роевых алгоритмов, указаны существующие трудности в применении алгоритмов роевого интеллекта.

## 1.1. История развития теории оптимизации

Корни методов оптимизации тянутся еще ко времени зарождения первых цивилизаций. Распределение ресурсов, планирование работ, организация маршрутов и многие другие проблемы были актуальны на протяжении всей истории человечества, но математические модели для их формального описания и методы решения возникли относительно недавно. Некоторые из задач, ставших впоследствии классическими, формулировались уже в древности в форме легенд. Так, например, древнегреческая легенда о Дидоне, финикийской царице, стала подспорьем для возникновения задачи об охвате максимальной площади нитью конечной длины [Зельдович, 1972]. В конечном итоге эта и ряд подобных задач заложили основы того, что в современной математике стало принято называть вариационным исчислением. В XVII веке ученые столкнулись с задачами, где наилучшие решения зависят не только от управляемой переменной, но и от выбора функции в целом. Позднее И. Бернулли в 1696 году, подмечая специфику задач такого рода, предлагает математикам задачу о брахистохроне и, по совету Лейбница, на конкурсной основе [Петров, 2005]. В последующие годы многими математиками решались отдельные частные случаи вариационных задач, но честь

открытия общего метода решения вариационных задач принадлежит Леонарду Эйлеру. Позднее метод Эйлера был усовершенствован Л. Лагранжем.

Эпоху развития вариационного исчисления можно назвать также эпохой развития методов непрерывной оптимизации. В то же время развивались и дискретные методы, развитие которых являлось следствием удачного разрешения многих характерных для задач комбинаторной или дискретной оптимизации. Так, в работе [Schrijver, 2005] выделяется семь типов задач дискретной оптимизации и приводится их историческое описание до 1960 года. Было бы неверным утверждать, что дискретная математика появилась в какой-то определенный момент истории, однако зарождение таковой чаще всего связывают с именами Г. Лейбница и Л. Эйлера [Ватутин, 2016]. Первого называют отцом комбинаторики [Белл, 1979], а работы второго положили начало развитию теории графов. Широкое распространение задачи дискретной оптимизации начали получать с конца XIX – начала XX века, когда прежде чисто умозрительные конструкции стали получать применение в различных областях экономики.

Как и во многих науках XX век и в области оптимизации явился очень плодотворным столетием. В эту эпоху пополнение богатства настоящего предмета связано с именами Д. Неймана, Л. Канторовича, Д. Данцига – основоположниками линейного программирования; Р. Беллмана – динамического программирования; Б. Эгервари, А. Колмогорова, К. Шеннона, и многих других. Некоторое пресыщение математической действительности суровыми абстрактными моделями не могло не привести к появлению математических моделей, опирающихся на закономерности, взятые из других дисциплин. Речь идет об адаптивных алгоритмах оптимизации, имитирующих поведение «живых» или физических систем. В работе [Гараничин, 2003] дается обоснование утверждению, что предпосылками появления таких алгоритмов являются фундаментальная работа Н. Винера «Кибернетика» [Винер, 1983], а также рост вычислительных мощностей. Подробный и глубокий исторический обзор методов оптимизации приведен в монографии [Weise, 2009].

Отдельно необходимо указать развитие методов локального поиска в решении задач оптимизации. Первые алгоритмы локального поиска для решения

задач дискретной оптимизации описываются в работах конца 50-х, начала 60-х годов XX века. Затем до середины 80-х годов развитие этого направления приостановилось, поскольку данные алгоритмы не позволяли найти глобальный экстремум задачи и часто даже не давали достаточное приближение к нему. Новый этап развития методов локального поиска связан с применением эвристик, основанных на аналогиях с природными процессами, такими как формирование кристаллической решетки, естественный отбор, коллективное поведение насекомых, бактерий, птиц. Важным отличием таких методов является стохастичность и допуск ухудшений рассматриваемых решений на отдельных итерациях. Многие современные методы локального поиска могут быть описаны с помощью конечных цепей Маркова, что позволяет исследовать их вероятностную сходимость к глобальному оптимуму [Кочетов, 2000].

К наиболее эффективным алгоритмам локального поиска можно отнести следующие:

- поиск с запретами;
- имитация отжига;
- генетические (эволюционные) алгоритмы;
- роевой интеллект.

Генетические и роевые алгоритмы роевого интеллекта объединены в класс популяционных алгоритмов, названных так по причине использования популяции решений, то есть множества альтернативных решений задачи, которые взаимодействую между собой определенным образом, что и обеспечивает выполнение локального поиска. Краткая история данных алгоритмов и их особенности рассмотрены в соответствующих пунктах подраздела 1.2.

**Классификация методов оптимизации**. Существует множество способов классифицировать методы оптимизации в зависимости от различных признаков. Одним из важнейших признаков является наличие в методе стохастичности. По этому признаку методы можно разбить на три большие категории по критерию использования элемента случайности:

- детерминированные;

- случайные (стохастические).

В работе [Жилинскас, 1989] отдельно выделяются комбинированные методы, то есть имеющие признаки как детерминированных, так и стохастических методов. Однако наличие стохастических свойств позволяет однозначно провести разделение на два класса, а формальную границу межу стохастическими и комбинированными методами провести очень сложно. На рисунке 1.1. приведены примеры из работы [Weise, 2009].

Методы оптимизации могут также классифицироваться и в соответствии с задачами оптимизации. В работе [Biegler, 2004] задачи оптимизации классифицируются по критерию типу множества управляемых переменных $X$: непрерывному или дискретному (рисунок 1.2):

- ЦЛП – целочисленное линейное программирование;
- НЛП – Нелинейное программирование;
- КП – квадратическое программирование;
- ИО – имитация отжига;
- ГА – генетические алгоритмы.

По критерию гладкости и наличия у функции критерия оптимальности частных производных оптимизационные методы можно разбить на три класса:

- прямые методы не требуют ничего кроме вычислений критерия в точках приближений;
- методы 1-го порядка требуют вычисления первых частных производных функции критерия;
- методы 2-го порядка требуют вычисления вторых частных производных функции критерия.

Повсеместное употребление понятия оптимизация, пусть даже в математическом смысле, породило огромное множество различных видов задач, которые будут разниться в зависимости от области науки, которая тесно взаимодействует с оптимизацией.

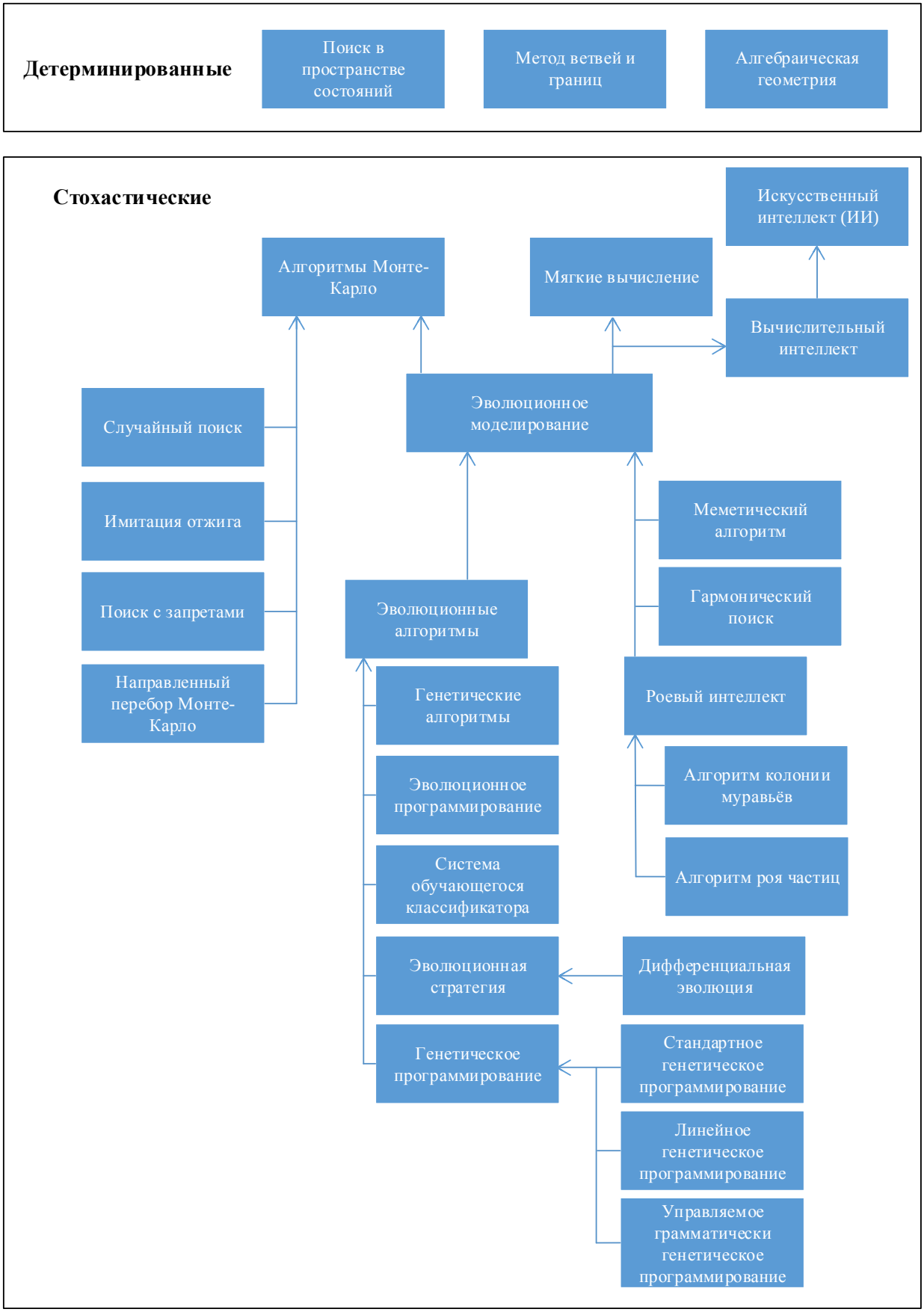



Рисунок 1.1 – Классификация алгоритмов оптимизации по критерию использования стохастичности

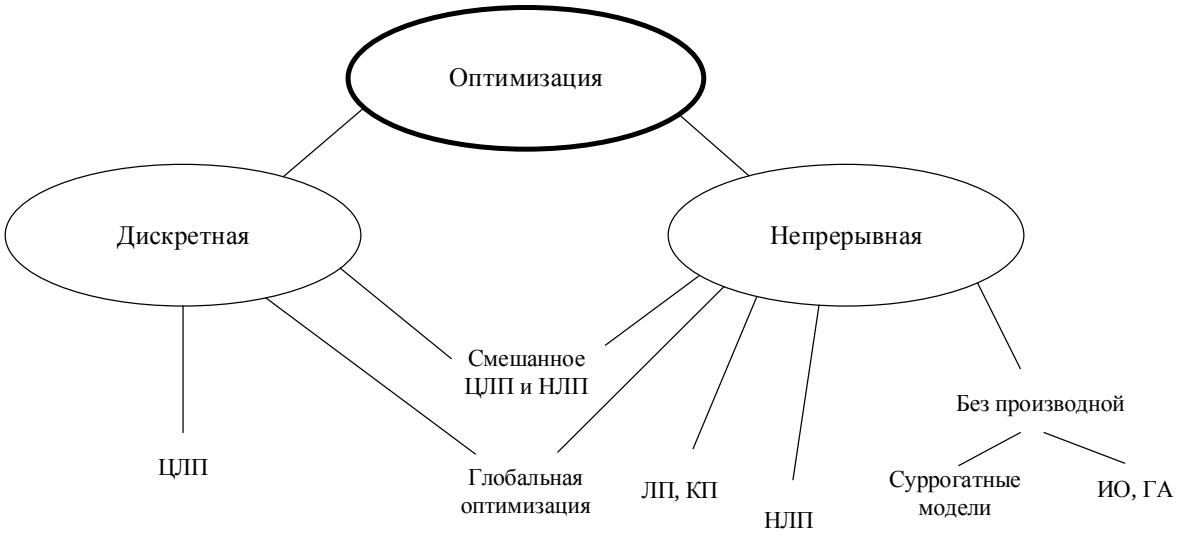



Рисунок 1.2 – Классификация задач оптимизации и методов решения по свойствам управляемых переменных

1. Задачи назначения – их разновидностью могут быть задачи сетевого планирования и управления, когда необходимо задействовать, к примеру, работников предприятия так, чтобы минимизировать общее время выполнения работы.

2. Транспортные задачи – также известны как задачи планирования перевозок.

3. Задачи отыскания максимального потока – включает в себя задачу массового обслуживания, когда необходимо найти оптимальное количество терминалов для обслуживания клиентов в данный момент времени, дабы обеспечить максимальный поток клиентов и отсутствие простаивающих терминалов.

4. Отыскание минимального остовного дерева – в качестве примера можно привести задачу отыскания такой сети автодорог между всеми городами в регионе, чтобы стоимость строительства такой дороги была минимальна. Похожие задачи могут встречаться в энергетике, сетях коммуникации и т. д.

5. Отыскание кратчайшего пути – задача поиска оптимального пути согласно заданному критерию. Задачи подобного класса решаются в GPS-навигаторах, для

отыскания оптимальной последовательности операций, ведущей к решению комбинаторной задачи, в области телекоммуникации для наискорейшей передачи пакета данных по сети.

6. Задача коммивояжера – она же «задача выбора маршрута». В отличие от задачи поиска кратчайшего пути, здесь необходимо, как правило, обойти все вершины данного графа оптимальным образом по критерию длительности пути и/или стоимости затрат на дорогу и др.

В заключение обзора схем классификации оптимизационных методов приведем классификацию из [Biegler, 2004], где автор приводит не только обширный спектр задач, но и указывает методы, которые могут быть применены к тому или иному типу задач (Таблица 1.1). В настоящее время многие задачи оптимального проектирования и управления техническими системами могут быть сведены к указанным в таблице 1.1 классам.

Задачи проектирования и управления техническими системами часто обладают следующими особенностями, которые приводят к значительной сложности их решения. Конечно, не каждая задача обладает всеми указанными свойствами.

1. Нелинейность. Как правило, в рассматриваемых задачах и критерий оптимальности, и ограничения, являются нелинейными функциями в силу нелинейности физических законов, определяющих зависимости в системе.

2. Высокая размерность. В технических системах присутствует большое количество варьируемых управляемых переменных и большое количество альтернатив как на этапе проектирования, так и во время управления. При наличии $n$ управляемых переменных, каждая из которых может принимать $m$ значений, общее число возможных решений составляет $m^n$.

3. Множество локальных экстремумов. Нелинейные зависимости и большое количество альтернатив приводит к большому количеству локальных экстремумов. Поэтому существует множество решений, локальное изменение которых не приводит к улучшению, в результате процесс оптимизации может остановиться в локальном решении задачи.

Таблица 1.1 – Классы оптимизационных задач с методами решения

| Область применения | ЛП | Смешанное ЦЛП и ЛП | КП и выпуклое | НЛП | Смешанное ЦЛП и НЛП | Глобальная Опт-ия | ИО/ГА |
|---|---|---|---|---|---|---|---|
| **Конструирование и синтез** | | | | | | | |
| Синтез теплообменных сетей | x | x | | x | x | x | x |
| Синтез массообменных сетей | x | x | | x | x | x | x |
| Сортировщики | | x | | | x | | |
| Регуляторы | x | | | x | x | x | |
| Конструирование оборудования | | | | x | x | | x |
| Технологический процесс | | | | x | x | | |
| **Операции** | | | | | | | |
| Планирование | x | x | | | x | | x |
| Распределение ресурсов | x | x | | | x | | |
| Оптимизация в реальном времени | x | | x | x | | | |
| **Управление** | | | | | | | |
| Линейные модели управления | x | | x | | | | |
| Нелинейные модели управления | | | | x | | x | |
| Гибридные модели управления | | x | | x | x | | |

4. Недифференцируемость. Многие рассматриваемые задачи не имеют аналитического выражения для функции критерия оптимальности, которое можно было бы исследовать с помощью методов дифференциального исчисления. Это относится не только к задачам комбинаторной оптимизации, но и к другим задачам, в которых модель оптимизируемого объекта рассчитывается алгоритмически.

5. Многофакторность. Как правило, для повышения технико-экономических показателей функционирования технических систем необходимо рассмотрение большого числа возможных величин, на которые можно оказать влияние (управлять которым). В результате для достижения максимальной эффективности требуется решать задачи не только параметрической, но структурной оптимизации, рассматривать большое количество различных управляемых переменных.

6. Многокритериальность. Технические системы могут выполнять несколько функций, каждая из которых может иметь свои критерии эффективности. И кроме критериев, показывающих насколько успешно система выполняет свои функции, важны критерии, связанные с потреблением материальных ресурсов и затратами времени.

7. Необходимость решать в реальном времени. Данное свойство не относится к задаче проектирования, только управления. Скорость, с которой система реагирует на изменения окружающей среды, важна в первую очередь для безопасности и экономичности.

## 1.2. Обзор методов оптимизации

### 1.2.1. Детерминированные методы

**Полный перебор** или метод «грубой силы» (от англ. Brute force) является наиболее простым методом решения комбинаторных оптимизационных задач, путем непосредственного перебора всех возможных решений задачи. Трудоемкость метода полного перебора прямо зависит от числа комбинаций, поэтому в задачах высокой размерности решение данным методом может выполняться в течение дней, лет и даже дольше. Например, если требуется найти оптимальную комбинацию значений $n$ переменных, каждая из которых может принимать $m$ значений, то придется проанализировать $m^n$ комбинаций. Для несложной на первый взгляд задачи с $n = 16$ и $m = 8$ число комбинаций составит $2{,}815 \cdot 10^{14}$ вариантов. При обработке миллиона вариантов в секунду потребуется 9 лет для полного перебора.

При этом существует класс задач, в основном криптографических, где полный перебор является на данный момент единственным возможным для использования. Например, время подбора простейшего пароля, состоящего из $n$ битов, пропорционально $2^n$ при отсутствии дополнительных данных о пароле [Кормен, 2005]. Как показывается в работах [Chvátal, 1972; Кормен, 2005; Скиена, 2013], метод полного перебора является единственным известным методом решения или эффективным вспомогательным средством для решения таких задач, как обнаружение коллизий хеш-функций и поиск ближайшего соседа.

Для задач, в которых качество отдельного решения проверить можно очень быстро и размерность которых невелика, полный перебор может быть эффективным средством решения при использовании распараллеливания расчетов. При этом распараллеливание полного перебора, несмотря на кажущуюся простоту, в некоторых случаях может оказаться нетривиальной задачей [Родионов, 2014].

**Метод ветвей и границ.** Очевидным направлением совершенствования метода полного перебора является отсечение части заведомо неэффективных вариантов. В 1960 году Ленд и Доиг в работе [Land, 1960] предложили новый метод решения задач дискретной комбинаторной оптимизации. Метод можно кратко охарактеризовать как перебор решений с отсевом подмножеств, заведомо не содержащих оптимальный результат. Формальное описание алгоритма для решения задач минимизации может быть записано следующим образом [Гончаров, 2005]:

Пусть рассматривается задача вида:

$$f(X) \rightarrow min,\ X \in D,$$

где $f(X)$ – критерий оптимальности (целевая функция), $D$ – конечное множество допустимых решений или область поиска решений.

Пусть имеется некоторое множество решений $d \subseteq D$. Функцию $b(d)$, ставящую в соответствие $d$ его разбиение на более одного подмножества $d_1, ..., d_n$, называют ветвлением. Функция $H(d)$, такая что $H(d) \leq \min f(X)$; $X \in d$, называется нижней границей функции.

На каждом шаге алгоритма имеется наилучшее на данный момент решение задачи (рекордом) $X_0$ и подмножества $t_1, t_2, ..., t_L$ еще не проверенных решений. На первом шаге $L = 1$, $t_1 = D$, $X_0$ – случайно выбранный элемент из множества $D$ или же просто пустое множество. Шаг начинается с проверки еще не просмотренных решений. Множество $t_j$ отсекается при выполнении хотя бы одного из условий:

- $H(t_j) \geq f(X_0)$;

- $H(t_j) < f(X_0)$ и найдено $y_j \in t_j$, для которого $f(y_j) = \min f(X) = H(t_j)$; $X \in t_j$.

Во втором случае найдено новое наилучшее решение и выполняется смена рекорда $X_0 = y_j$. Пусть $t_1, t_2, ... , t_M$ ($M \leq L$) – не отсеченные множества (отсечены множества с номерами $M + 1, ..., L$). Если $M = 0$, то $X_0$ – решение задачи, алгоритм завершается. Иначе среди $t_1, ..., t_M$ выбирается множество для следующего ветвления $b(t_1) = (d_1, ..., d_n)$, дающего множества $d_1, ..., d_n, t_2, ... , t_M$, которые нумеруются последовательно от 1 до $L$. На этом шаг завершается и начинается выполнение следующего шага.

Наиболее часто в литературе встречается применение данного алгоритма для решения задачи коммивояжера, поскольку весьма удобной оказывается графическая интерпретация алгоритма. В общем случае метод может быть применен для различных задач целочисленного линейного программирования. Время решения задач оптимизации методом ветвей и границ в общем случае экспоненциально возрастает с увеличением размерности задачи, что делает применение метода ограниченным на практике. Для каждого класса задач следует реализовывать особые правила ветвления и определения границ, что привод к сильной зависимости эффективности метода от опыта и умений автора конкретной реализации. В наихудшем случае алгоритм, реализующий метод ветвей и границ, может свестись к полному перебору. Достоинствами метода являются универсальность, гарантированное получение наилучшего решения, простота реализации на ЭВМ. Последнее свойство, однако, зависит от решаемой задачи, не всегда можно легко определить правила ветвления.

**Поиск с возвратом.** Термин «backtracking» (поиск с возвратом или перебор с возвратом) был введен в 1950 году американским математиком Дерриком Генри

Лемером [Freuder, 2006]. Говорить о чьем-то приоритете в разработке данного метода некорректно ввиду его очевидности. Суть поиска с возвратом состоит в расширении ранее сформированного частичного решения. Если расширение текущего частичного решения невозможно, то алгоритм возвращается к одному из прошлых ранее найденных решений для его расширения. Наилучшим образом продемонстрировать данный метод можно на примере решения задачи о размещении 8 ферзей: расположить 8 ферзей на шахматной доске так, чтобы ни один ферзь не мог быть побит другим ферзем. Первый ферзь размещается на некоторой клетке доски, после чего начинается размещение других. Если дальнейшее размещение ферзей невозможно так, чтобы один не бил другого, то происходит возврат на шаг назад, и предыдущий ферзь меняет положение.

Данный метод имеет прямую связь с методом ветвей и границ, поскольку тот является, в сущности, его расширенным дополнением. Другие модификации «бэктрэкинга» можно встретить под названиями: метод поиска в глубину, метод проб и ошибок и др.

Поиск с возвратом может быть реализован и как стохастический метод. Для этого процедура выбора нового решения делается стохастической. Однако возникает проблема выбора параметров выбора нового решения. Так, если максимальный шаг перехода будет велик, то алгоритм может оказаться в локальном экстремуме и не выйти из него. Если же шаг мал, то алгоритм будет работать слишком долго. Далее в классе стохастических методов будет рассмотрен метод имитации отжига, который способен выходить из локальных экстремумов.

**Градиентные методы.** Класс методов, использующих градиенты целевой функции для нахождения ее оптимума, является весьма распространенным из-за их простоты реализации и доступности для понимания; чаще всего применяются для решения задач нелинейного программирования. Несмотря свою простоту, алгоритмы, построенные по данному методу, имеют существенный недостаток, а именно: единожды достигнув локального экстремума, они уже оказываются неспособными выбраться из него. Ярким представителем данного класса является

метод градиентного спуска, который весьма коротко можно охарактеризовать рекуррентной формулой следующего вида:

$$X^{k+1} = X^k - \alpha \cdot \nabla f(X^k),$$

где α – коэффициент скорости при градиенте функции $f(X)$. К сожалению, столь простая реализация не лишена недостатков, а именно: плохие условия сходимости и малая скорость сходимости. Этих недостатков можно избежать при помощи некоторых модификаций данного алгоритма, например, модифицированный алгоритм Ньютона (или Ньютона – Рафсона) с рекуррентной формулой:

$$X^{k+1} = X^k - \alpha^k \cdot \nabla^2 f(X^k)^{-1} \cdot \nabla f(X^k),$$

который имеет лучшие условия и скорость сходимости благодаря учету второй производной. Также известны различные модификации данного метода. Алгоритм Левенберга-Марквардта, который был открыт авторами независимо друг от друга [Levenberg, 1944; Marquardt 1963], и является по сути объединением алгоритмов Ньютона-Рафсона и градиентного спуска.

Модификация, известная под названием алгоритма обратного распространения ошибки, пользуется большой популярностью в машинном обучении для обучения нейронных сетей, [Pedersen 2010a; Бураков, 2013].

**Симплекс-метод** (или алгоритм Данцига) – популярный алгоритм решения оптимизационных задач линейного программирования путем перебора вершин выпуклого многогранника в многомерном пространстве. Алгоритм получил свою популярность во многом благодаря универсальности его применения к самому широкому спектру задач, а также относительной простоте реализации и невысокому уровню требований к вычислительной мощности.

Приведем алгоритм решения задачи симплекс-методом путем составления симплекс-таблиц [Сидоркин, 2015; Оптимизация, 2017]. Пусть $x_j$ – свободные переменные; $y_i$ – базисные переменные ($i = 1 \ldots n$); $B_i$ – свободные члены; $a_{ij}$ – коэффициенты при свободных переменных в системе уравнений ограничений вида ($j = 1 \ldots m$):

$$y_i = b_i + a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ij}x_j + \cdots a_{im}x_m$$

Пусть $L$ – критерий оптимальности:

$$L = c_0 + \gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_j x_j + \cdots \gamma_m x_m \rightarrow max$$

При использовании симплекс-метода задача должна быть приведена к стандартной форме записи уравнений-ограничений и критерия и составлена симплекс-таблица путем записи свободных членов и коэффициентов в строки, с соответствующим номером *i*.

В ходе решения задачи на каждом шаге расчета выполняется замена одной свободной переменной на одну базисную, при этом коэффициенты симплекс-таблицы пересчитываются в соответствии с алгоритмом их преобразования.

**Метод множителей Лагранжа.** Ранее в разделе 1 уже упоминалось о значительном достижении Л. Лагранжа в начале XVIII века, который на основе работ Л. Эйлера создал передовой для его времени метод оптимизации, который позже получил его имя. Данный метод в отличие от предыдущего подходит как для задач линейного программирования, так и для нелинейного, а также широко применяется в различных задачах экономики, теории управления и энергетики. Ниже приводится описание метода [Акулич, 1986]:

Пусть $\varphi_i(X)$ есть *i*-ое из *m* ограничений. Составим функцию Лагранжа в виде линейной комбинации функции $f(X)$ и функций $\varphi_i(X)$ взятых с коэффициентами, называемыми множителями Лагранжа – $\lambda_i$:

$$L(X, \lambda) = f(X) + \sum_{i=1}^{m} \lambda_i \varphi_i(X).$$

Составим систему из *n*+*m* уравнений, приравняв к нулю частные производные функции $L(X, \lambda)$ по $x_j$ и $\lambda_i$.

Если полученная система имеет решение относительно параметров $x_j$' и $\lambda_i$', то *X*' может быть условным экстремумом, то есть решением исходной задачи. Проверить экстремальность точек в общем случае можно с помощью вычисления вторых производных функции Лагранжа *L*. Позднее метод Лагранжа был обобщен Куном и Таккером в работе [Kuhn, 2014], где были впервые найдены необходимые условия решения задачи нелинейного программирования в общем виде.

**Метод динамического программирования.** Термин «динамическое программирование» был впервые употреблен Ричардом Беллманом в 1940 г. для

описания процесса решения нахождения решения задачи, где ответ на одну задачу может быть получен только после решения предшествующей ей. Позднее термин был окончательно переопределен в 1954 г. [Bellman, 1954].

Рассмотренные выше методы линейного и нелинейного программирования можно охарактеризовать как одноэтапные потому, что решение в них находится в пределах выполнения одного алгоритма или, говоря иначе, за один шаг. Метод динамического программирования рассматривает поэтапное решение отдельных подзадач. Нахождение решения конкретных задач методами динамического программирования включает несколько этапов, на каждом из которых отыскивается решение частной задачи, являющейся составной частью исходной большой задачи. Принцип динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений. Приведем наиболее общий подход к решению некоторой задачи динамического программирования [Акулич, 1986].

Предположим, что данная физическая система $S$ находится в некотором начальном состоянии $S_0$ и является управляемой. В результате управления $X$ система переходит из заданного начального состояния $S_0$ в определенное конечное состояние из множества возможных $S_{кон}$. Качество управления $X$ оценивается с помощью критерия $f(X)$. Необходимо выбрать из множества возможных управлений такое, при котором критерий $f(X)$ принимает наилучшее (минимальное или максимальное) значение.

Классическими задачами динамического программирования являются:

- задача о наибольшей общей подпоследовательности;
- задача о вычислении чисел Фибоначчи;
- задача о порядке перемножения матриц;
- задача о ранце;
- максимальное независимое множество вершин в дереве.

**Жадные эвристические методы.** Помимо строго определенных алгоритмов, эффективность который обычно доказывается математически, в практике часто имеют место быть так называемые жадные алгоритмы.

Основой жадных эвристических алгоритмов является последовательное применение локально оптимального выбора. Такие алгоритмы эффективны в задачах, в самой природе которых заложено, что последовательность локально оптимальных выборов приводит глобально оптимальное решение [Скиена, 2013]. Например, алгоритм Дейкстры [Dijkstra, 1959] нахождения кратчайшего пути в графе является жадным, потому что на каждом шагу ищут вершину с наименьшим весом, в которой еще не были, после чего обновляют значения других вершин. Такое поведение называют эвристическим, то есть выбранный метод не обязательно гарантирует нахождения оптимального решения, но принимается допустимым в условиях конкретной задачи для сокращения, например, времени решения. Например, в непрерывной задаче о рюкзаке, действуя согласно жадному алгоритму, необходимо каждый раз выбирать предмет с наибольшей удельной стоимостью среди тех, которые еще не загружены. В такой задаче жадный алгоритм приводит к получение оптимального решения, но уже для дискретной задачи о рюкзаке жадный алгоритм может приводить как к оптимальному, так и к решению, которое очень далеко от оптимального [Скиена, 2013].

Следует понимать, что не существует общего подхода к формулированию жадного алгоритма и для каждой конкретной задачи требуется сформулировать свой. Однако существует достаточно универсальный способ определить приведет ли данный конкретный жадный алгоритм к оптимальному решению в условиях конкретной задачи. Для этого необходимо выяснить являются ли подмножества элементов данной задачи матроидом [Edmonds, 1971]. Если это выполняется, то согласно теореме Радо-Эдмондса [Rado, 1942] к задаче может быть применен жадный алгоритм, приводящий к оптимальному решению. Наиболее общим образом сформулировать жадный алгоритм можно так:

Пусть $X$ – носитель матроида (множество взвешенных элементов), а $I$ – семейство независимых подмножеств $X$.

Для всех $i$ от 0 до $|X|$ строится множество $A_i \in I$ мощности $i$, вес которого является минимальным среди весов независимых подмножеств той же мощности.

$A_0 = \emptyset$ на нулевой итерации.

Строятся множества следующим образом: $A_i = A_{i-1} + \{x\}$, где $x$ – минимальный из элементов $y \in X \backslash A_i$ таких, что $A_i \cup \{y\} \in I$.

Ответ задачи – $A_n$, где $n$ – ранг $X$.

Преимуществом жадных алгоритмов является очень высокая скорость работы, так как на решение формируется только один раз и на каждом шаге выполняется только выбор наилучшего шага без учета результатов последующих шагов. Важно отметить, что во многих оптимизационных задачах эффективность жадных алгоритмов значительно зависит от выбора начального решения и условий решаемой задачи [Кормен, 2005; Скиена, 2013]. Жадные алгоритмы могут иметь и стохастические свойства, в этом случае они переходят из класса детерминированных в класс стохастических алгоритмов. Для отдельных вероятностных жадных методов может быть доказана асимптотическая сходимость к глобальному оптимуму задачи независимо от начальных условий [Кочетов, 2000].

### 1.2.2. Стохастические методы

**Случайный поиск.** Наиболее простым методом можно считать произвольную случайную генерация решений без какого-либо учета качества ранее проверенных решений с последующим выбором среди них наилучшего. В работе [Скиена, 2013] указаны условия, при которых случайный поиск можно рассматривать как эффективный способ решения задач.

1. В пространстве поиска решений имеется высокая доля решений с удовлетворительным качеством. Например, поиск больших простых чисел.

2. Топология пространства поиска решений такова, что по результатам оценки ранее полученных решений невозможно сделать определенные выводы о направлении изменения значения критерия, даже в узкой локальной окрестности. Простейший пример – подбор ключа к шифру, когда обратной связью является только «подходит / не подходит».

Для большинства задач оптимизации метод произвольной выборки не позволяет получить хороших результатов, так как такие задачи имеют небольшое

количество хороших решений и высокую степень однородности [Растригин, 1968; Скиена, 2013]. Главным недостатком метода является неиспользование результатов, полученных на предыдущих шагах. Поэтому существует большое множество модификаций метода, использующих различные эристики. В качестве одного из видных деятелей в области методов случайного поиска можно выделить Л. А. Растригина [Растригин, 1968]. Также модификации методов случайного поиска детально описаны в работе [Malek, 1989].

**Поиск с возвратом** является аналогом приведенного выше детерминированного алгоритма, в котором, вместо последовательного перебора вариантов, используется генерация псевдослучайного вектора значений переменных оптимизируемой функции.

*Алгоритм с парной пробой*. Представляет собой алгоритм случайного поиска, в котором на каждом последующем шаге выбирается значение по обе стороны от текущего значения. То есть если на $k$ – ом шаге, вектор управляемых переменных параметров был равен $X_k$, то на $k$+1 шаге $X_{k+1} = X_k \pm \lambda\xi$, где $\xi$ – случайное решение; $\lambda$ – величина пробного шага, которая определяет направление следующего шага алгоритма, вычисленное исходя из наилучшего предыдущего значения.

*Алгоритм наилучшей пробы.* На исходной точке делается $m$ случайных шагов и запоминается тот шаг, который привел к наименьшему значению критерия. Рабочий шаг делается именно в этом направлении. С увеличением числа $m$ случайных шагов выбранное направление все более приближается к направлению антиградиента.

*Алгоритм статистического градиента.* Из исходного состояния делается $m$ случайных шагов. На основании этих результатов высчитывается векторная сумма, в пределе при $m \to \infty$ приближающаяся к направлению градиента. Особенностью метода статистического градиента является возможность принятия решений о направлении рабочего шага при заданном значении $m$. В то же время обычное численное определение градиента требует не менее n пробных шагов.

Модификации повышают эффективность случайного поиска, но вместе с тем увеличивают его сложность, поскольку возникает задача настройки эвристических

правил, и вероятностных параметров работы метода. Для этой задачи требуется анализ особенностей задачи, многочисленные эксперименты, исследование эффективности различных значений параметров алгоритма. Может быть использовано и динамическая автонастройка, особенности которой подробно изучены в работе [Растригин, 1969]. Однако и автоматическую настройку необходимо правильным образом реализовать и ее эффективность может быть различной в зависимости от класса и размерности решаемых задач.

**Поиск с запретами.** В 1986 году американский ученый Ф. Гловер в работе [Glover, 1986] представляет миру принцип поиска оптимальных решений, названный поиском с запретами (от англ. Tabu search), который им же формализуется в 1989–90 гг. в работах [Glover, 1989; Glover, 1990]. Данный принцип предоставляет возможность процессу поиска не останавливаться в локальном оптимуме, а перемещаться между оптимумами, в попытках найти среди них глобальный. Это достигается при помощи так называемого эффекта памяти, который позволяет запоминать оптимальные решения, полученные на предыдущих шагах алгоритма и вносить их в список запретов $Tabu(X_k)$. Запрещая возвращаться к оптимальному решению, алгоритм также запрещает некоторую окрестность этого решения, что позволяет выходить из локального максимума в поисках другого.

Существует множество модификаций поиска с запретами [Glover, 2013]. Рассмотрим стохастическую окрестность $N_p(X) \subset N(X)$, где все элементы окрестности $J \in N(X)$ включаются в множество $N_p(X)$ с некоторой вероятностью $p$. Вероятность не зависит от других элементов. В предельных случаях $N_p(X)$ может совпадать с $N(X)$ или наоборот быть пустым. Шаги алгоритма можно записать следующим образом:

1. Выбрать начальное решение из множества допустимых решений $X_0 \in D$ и положить

$f^* = f(X_0)$, $Tabu(X_0) = \emptyset$, $k = 0$.

2. Пока не выполнен критерий остановки:

2.1. Сформировать окрестность $N_p(X_k)$.

2.2. Если $N_p(X_k) = \emptyset$, то $X_{k+1} = X_k$, иначе найти $X_{k+1}$ такой, что

$$f(X_{k+1}) = \min \{f(J) \mid J \in N_p(X_k) \setminus Tabu(X_k)\}.$$

2.3. Если $f^* > f(X_{k+1})$, то $f^* = f(X_{k+1})$.

2.4. Положить $k = k+1$ и обновить список запретов $Tabu(X_k)$.

Параметры $p$ и $l$, (где $l$ – длина списка запретов) являются управляющими для данного алгоритма и выбор их численных значений зависит как от размерности задачи, так и от мощности окрестности. По значению параметра $l$, можно условно выделить три типа памяти: краткосрочную, среднесрочную и долгосрочную [Malek, 1989]. Как следует из их наименований, выбор того или иного типа памяти зависит от конкретной задачи. Например, как сказано в [Malek, 1989], одной лишь краткосрочной памяти может быть достаточно, чтобы достичь решения, превосходящего те, которые были найдены обычными локальными методами поиска, но среднесрочная и долгосрочная память часто необходимы для решения сложных задач. При выполнении определенных условий алгоритм асимптотически сходится к глобальному экстремуму задачи [Кочетов, 2000].

**Алгоритм имитации отжига** был создан в 1980-е годы, его авторы: Скотт Киркпатрик, Даниель Желатт, Марио Вечи и Владо Церни [Kirkpatrick, 1983]. Может считаться особой версией алгоритма Метрополиса [Metropolis, 1953]. Алгоритм основан на использовании аналогий с процессом рекристаллизации вещества, в ходе которого вещество охлаждается и отвердевает, а скорость движения его молекул падает.

Классический алгоритм можно описать следующим образом [Бураков, 2013; Скиена 2013]:

1. Температура получает максимальное значение: $T = T_{max}$. Номер итерации $k = 1$.

2. Выбирается начальная точка (решение оптимизационной задачи) $X = X_0$.

3. Рассчитывается критерий $f(X)$.

4. Аргумент получает случайное приращение, которое генерируется согласно выбранной схемы реализации алгоритма $X' = X + \Delta X$.

5. Рассчитывается критерий $f(X')$.

6. Рассчитывается изменение критерия $\Delta f = f(X') - f(X)$.

7. Если $\Delta f < 0$, то $X = X'$ (выполняется переход в новое состояние, то есть к новому решению).

8. Если $\Delta f > 0$, то вероятность перехода в новое состояние (решение $X'$) вычисляется по формуле:

$$p(\Delta f) = -\exp\left(\frac{\Delta f}{kT}\right)$$

где $k$ – константа, регулирующая быстроту сходимости алгоритма.

Затем $p(\Delta f)$ сравнивается со случайным числом $n \in [0,1]$.

Если $p(\Delta f) > n$, то $X = X'$, иначе $X$ не изменяется.

9. Температура $T$ уменьшается, происходит возврат к шагу 4, и так до тех пор, пока $T$ не уменьшится до заданного порогового значения. Формула изменения температуры различается в разных модификациях алгоритма, обычно имеет следующий вид:

$$T_k = \frac{T_k}{e^{k \cdot v_t}}$$

где $v_t$ – коэффициент, задающий быстроту понижения температуры.

10. Номер шага увеличивается: $k = k + 1$.

Таким образом, пока искусственная температура $T$ большая, алгоритм отжига может делать большие шаги даже в направлениях, увеличивающих значение минимизируемой функции. В результате этой особенности оказывается возможным попасть в окрестность глобального минимума. При этом важно правильно определить начальное значение $T_{max}$ и задать закон изменения температуры. Существует большое множество модификаций алгоритма имитации отжига, например, адаптивный отжиг [Скиена, 2013], алгоритм квантового отжига [Finnila, 1994]. Алгоритм имитации отжига обладает высокой эффективностью и быстродействием. Доказана его сходимость к глобальному экстремуму независимо от топологии задачи [Bertsimas, 1993; Кочетов, 2000]. Недостатком алгоритма является необходимость выбора зависимости температуры от номера итерации и вероятности перехода от температуры и эвристических параметров алгоритма. Без

такой настройки эффективность алгоритма может быть существенно ниже [Скиена, 2013].

**Эволюционные алгоритмы**. В 1954 году вышла работа Нильса Баричелли, посвященной компьютерному моделированию процессов естественного отбора [Barricelli, 1954]. Это послужило началом математического моделирования, основанному на эволюционных механизмах. Применения эволюционных алгоритмов для решения задач оптимизации получили широкую известность после публикации работ Инго Рехенберга и Ханса-Пауля Швефеля в 1960–70-х гг. В основе алгоритма лежит аналогия естественного отбора, но отбираются решения оптимизационной задачи, таким образом выполняется итерационная процедура локального поиска [Емельянов, 2003; Гладков, 2009; Кочетов, 2017]. Решение представляется в виде вектора значений, который называется хромосомой или особью, а множество таких решений на каждом шаге алгоритма называют популяцией. Решение-особь оценивается значением критерия оптимальности, который достигается на данном решении. Решения с лучшими значениями критерия отбираются в следующую итерацию и оказывают влияние на решения, который будут созданы на этой итерации (дочерняя популяций или потомство). Принципиальные моменты механизма эволюции в различных ее аспектах могут быть формализованы по-разному [Гладков, 2006; Батищев, 2007; Панченко, 2007]. Вследствие этого появляется огромное количество разновидностей эволюционных алгоритмов [Гладков, 2009; Карпенко, 2012]: алгоритм растущих деревьев; сорняковый алгоритм; гармонический поиск; генетический алгоритм; культурный алгоритм; кукушкин поиск; алгоритм эволюции разума.

Достоинством эволюционных алгоритмов является простота применения для различных задач оптимизации, недостатками – невысокая эффективность в решении задач, обладающих большой размерностью и сложной структурой. Кроме того, требуется выполнять большое количество расчетов критерия. Критики справедливо указывают и на низкую скорость работы, поскольку эволюция и в природе протекает медленно. Далее в работе будет использован генетический алгоритм, поэтому он подробнее описан ниже.

Как правило, ГА начинает работу с создания начальной популяции как стохастического конечного множества решений задачи оптимизации. На каждой итерации создается новая популяция решений путем селекции, скрещивания и мутации. Чтобы создать новую особь популяции, отбираются два или более родительских особей. При этом вероятность выбора некоторого решения как родительского пропорциональна значению критерия, полученному на этом решении. В ходе скрещивания по какому-либо правилу происходит обмен элементами хромосом (кроссинговером). Одним из наиболее простых правил является разделение двух хромосом на две части каждой и обмен этими частями, как показано на рисунке 1.3. Полученное кроссинговером решение с некоторой вероятностью мутирует, то есть применяется изменяющее значений, записанных в хромосоме (генов). Изменение может быть на случайную величину, которая прибавляется к текущему значению гена, может быть выполнена замена без учета текущего значения. Этапы отбора, скрещивания и мутации повторяются для новой популяции и так до выполнения условия остановки алгоритма. В каждом вновь полученном поколении возникают новые хорошие и плохие решения рассматриваемой задачи. За счет отбора лучших решений с большей вероятностью среднее качество популяции с точки зрения критерия постепенно возрастает. Часто в алгоритме определенное количество наилучших хромосом текущей популяции переходит в следующую популяцию без изменений. Как показано в работе [Кочетов, 2000] при сохранении хотя бы одного наилучшего решения ГА асимптотически сходится к глобальному экстремуму.

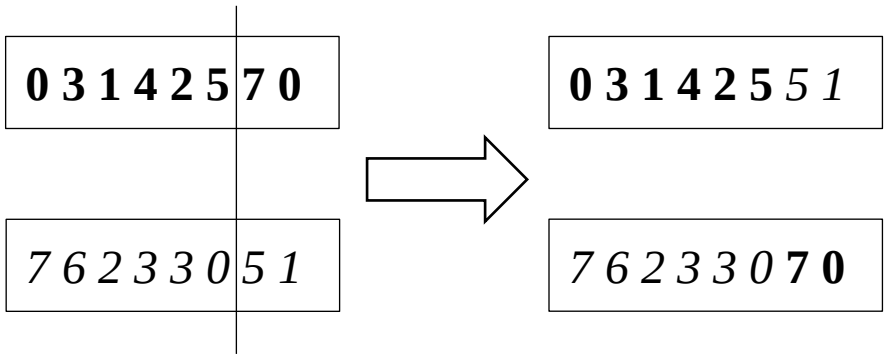



Рисунок 1.3 – Пример кроссинговера

## 1.3. Алгоритмы роевого интеллекта

### 1.3.1. История развития роевого интеллекта

Подобно тому как в природе образовалось коллективное поведение на основе сложного взаимодействия между элементами эволюционного отбора, так и в оптимизации вслед за эволюционными алгоритмами логичным образом появляются алгоритмы роевой оптимизации, которые объединились на сегодняшний день, образовав группу методов роевого интеллекта. В 1989 году появляется работа Херардо Бени и Ван Цзина, посвященная клеточным автоматам, в которой впервые фигурирует термин «роевой интеллект» (Swarm Intelligence) [Beni, 1993]. Можно выделить два алгоритма роевого интеллекта, которые появились раньше других и показали высокую эффективность РИ в решении оптимизационных задач: алгоритм роя частиц и муравьиный алгоритм. Авторы этих алгоритмов были вдохновлены моделями коллективного поведения в природе и смогли их применить для создания оптимизационных алгоритмов.

Одной из первых моделей, реализующих коллективное поведение, стала модель перемещения птиц в стае, созданная в 1986 году Крейгом Рейнольдсом [Reynolds, 1987]. Несмотря на простоту модели, она давала правдоподобную визуализацию коллективного поведения стаи. Дж. Кеннеди и Р. Эберхарт в 1995 году предложили алгоритм оптимизации непрерывных нелинейных функций. Авторы назвали его алгоритмом роя частиц (Particle Swarm Optimization – PSO) [Kennedy, 1995]. Они отталкивались от модели Рейнольдса, и близкой к ней работы Хеппнера и Гренадера [Heppner, 1990]. Авторы не напрямую скопировали особенность поведения птиц в стае, в своей работе Кеннеди и Эберхарт стремились достичь имитации социального поведения с помощью простых правил перемещения частиц. В настоящее время алгоритм роя частиц претерпел множество модификаций, которые в разное время формулировались различными учеными. Среди них можно выделить, например, модификации самого Эберхарта совместно с Юхи Ши [Shi, 1998a, Shi, 1999].

Вместе с тем развивалось и второе указанное направление, имеющее обширную историю, которая корнями уходит также в биологию. В 1959 году Пьер-Поль Грассе изложил теорию стигмергии [Dorigo, 1996], для объяснения поведения колонии термитов. Стигмергия – механизм спонтанного непрямого взаимодействия между индивидами, заключающийся в оставлении индивидами в окружающей среде меток, стимулирующих дальнейшую активность других индивидов. Как и в случае с птицами, сообщающими друг другу информацию о перемещении, постановка меток муравьями представляет аналог элемента, передающего информацию от одного агента алгоритма к другому. Формальное воплощение алгоритму дает Марко Дориго [Dorigo, 1996] в своих работах, начиная с опубликованной диссертации в 1992 году, что и является рождением данного алгоритма. Муравьиные алгоритмы показали высокую эффективность при решении комбинаторных задач и задач на графах [Штовба, 2003]. В дальнейшем было разработано множество модификаций алгоритма. Одной из наиболее эффективных модификаций является повышение уровня феромона на лучших найденных путях [Stützle, 1997].

Роевые алгоритмы основаны на итерационной процедуре локального поиска, могут быть описаны конечными цепями Маркова для исследования вероятностной сходимости к глобальному оптимуму задачи [Кочетов, 2000]. На сегодняшний день известны десятки роевых алгоритмов для решения задач стохастической оптимизации. В таблице 1.2 указаны наиболее известные из них.

### 1.3.2. Области применения алгоритмов роевого интеллекта

Благодаря своей гибкости и универсальности алгоритмы РИ используются во многих задачах проектирования и управления техническими системами [Poli, 2007; Зайцев, 2010; Thangaraj, 2011; Баранюк, 2015].

1. Построение телекоммуникационных систем алгоритмы РИ используются для проектирования структур сетей, управления потоками данных в сетях [Чураков, 2006; Zhang 2007; Zhao, 2010; Токшотов, 2016].

Таблица 1.2 – Перечень роевых алгоритмов

| **Название алгоритма** | **Ссылка на источник(и)** | **Автор(ы)** | **Год** |
|---|---|---|---|
| Алгоритм колонии муравьев | [Dorigo, 1996] | М. Дориго и В. Маньеззо | 1991 |
| Алгоритм роя частиц | [Kennedy, 1995] | Дж. Кеннеди и Р. Эберхарт | 1995 |
| Стохастический диффузионный поиск | [Bishop, 1989] | М. Бишоп | 1989 |
| Алгоритм роения бактерий | [Passino, 2002] | К. Пассино | 2002 |
| Электромагнитный поиск | [Birbil, 2003] | И. Бирбиль | 2003 |
| Алгоритм пчел | [Karaboga, 2005; Pham, 2005] | Д. Карабога / Д. Фам и др. | 2005 |
| Алгоритм динамики формирования рек | [Rabanal, 2007] | П.Рабанал, И.Родригес, Ф.Рубио | 2007 |
| Поиск косяком рыб | [Filho, 2008] | Б. Фило | 2008 |
| Обезьяний алгоритм | [Zhao, 2008] | Т. Ваншенг, Р. Джао | 2008 |
| Алгоритм интеллектуальных капель воды | [Hossein, 2008] | Хм.-Ш. Хосейни | 2008 |
| Алгоритм летучих мышей | [Yang, 2010b] | Син-Ше Янг | 2008 |
| Алгоритм гравитационного поиска | [Rashedi, 2009] | Э. Рашеди | 2009 |
| Алгоритм светлячков | [Yang, 2010a] | Син-Ше Янг | 2010 |
| Алгоритм стаи волков | [Wu, 2014] | Х.-Ш. Ву, Ф.-М. Жанг | 2014 |

2. Поиск оптимальных последовательностей и комбинаций (маршруты, календарные планы, задачи раскроя). Данные задачи обычно сводятся к NP-трудным задачам (модификациям задач коммивояжера, раскраски графов, покрытия, о клике, транспортных) [Alexandrov, 1999; Dorigo 2006; Lui 2007; Курейчик, 2010; Курейчик, 2011; Лебедев, 2010; Леванов, 2010; Фроловский, 2011; Лебедев, 2012; Матренин, 2013c; Кочегурова, 2014].

3. Алгоритмы РИ используются в различных областях для построения и оптимизации моделей в области машинного обучения, направленных на классификацию и кластеризацию объектов. Алгоритмы роевого интеллекта часто используются и для построения и настройки структуры и параметров классификаторов [Pedersen, 2010a; Матренин, 2016; Garcia-Galan, 2015; Ходашинский, 2017a; 2017b], в том числе отбора признаков, используемых затем для классификации [Анфилофьев, 2017].

4. Проектирование систем управления и оперативное автоматическое или автоматизированное управление: разработка контроллеров в различных областях, управление транспортными потоками, контроль работы двигателей и управление ими, проектирование и управление электроэнергетическими системами [Yoshida, 2000; Alrashidi, 2006; Lee, 2007; Mahadevan, 2010; Jamian, 2012; Dazahra 2016; Manusov, 2016; 2017a, Манусов, 2017a].

5. Отдельно можно выделить применение в биомедицинских технологиях для диагностики заболеваний, разработке медицинских препаратов, кластеризации генов, магнитоэнцефалографии, [Eberhart, 1999; Poli, 2007; Pedersen, 2010a; Самигулина, 2016; Сарин, 2017].

6. Другие области применения: компьютерная графика и визуализация, металлургия, робототехника, системы обеспечения безопасности, обработка сигналов и изображений [Beni, 1993; Poli, 2007], маркетинг [Никифорова, 2016].

### 1.3.3. Ограничения на применение роевых алгоритмов

В настоящее время еще не существует устоявшейся терминологии и классификации в области алгоритмом роевого интеллекта.

1. **Классификация**. Описанные методы можно разделить на группы по различным признакам, однако единой четкой классификации на настоящее время не существует, что приводит к путанице в терминах. Например, часто даже специалисты, знакомые с алгоритмами колонии муравьев, роя частиц, роя пчел, не используют объединяющий их термин «роевой интеллект». Хотя перечисленные примеры являются просто различными реализациями метода роевого интеллекта. Кроме того, существуют различные точки зрения на отношение между эволюционными и роевыми алгоритмами.

2. **Терминология**. Нет четкого разделения понятий «метод» и «алгоритм». В русскоязычной литературе принято использовать термины «метод роя частиц», но «генетический алгоритм»; «муравьиный алгоритм» и, одновременно, «метод колонии муравьев». Кроме того, некоторые понятия, общие по своему смыслу для

многих стохастических методов, такие как «фитнесс-функция», «миграция», используются только для некоторых отдельных алгоритмов, для других то же самое понятие обозначается другими терминами.

3. **Объяснение** стохастических методов неспециалистам, обусловленное непривычными для технических наук терминами, взятыми из описания природы: «феромон»; «популяция»; «мутация»; «приспособленность»; «гибель»; «рой» и другими. Кроме того, бывает трудно понять, как именно связать роевой алгоритм и оптимизационную задачу.

Основная сложность при использовании роевых алгоритмов заключается в их настройке и доработке под различные виды оптимизационных задач и выборе значений их параметров: [Shi, 1998; Bergh, 2001; Carlisle, 2001; Eberhart, 2001; Штовба, 2003; Чураков, 2006; Dorigo, 2006; Гладков, 2009; Зайцев, 2010; Курейчик, 2010; Pedersen, 2010a; Курейчик, 2011; Abraham, 2011; Карпенко, 2012; Лебедев, 2013; Manusov, 2017a].

Наибольшее число аналитических работ выполнено по алгоритму роя частиц, как следует из обзоров [Bergh, 2001; Trelea, 2003; Pedersen, 2010a]. Многие исследования рассматривали как поведенческие параметры влияют на сходимость частиц. Однако М. Педерсен [Pedersen, 2010a] показывает, что, во-первых, мнение, что производительность оптимизации и сходимость роя тесно связаны, является предвзятым, а сходимость оптимизации роя частиц к некоторой точке в пространстве поиска не гарантирует автоматически, что точка сходимости будет оптимумом решаемой задачи. Во-вторых, предполагается, что точка притяжения частиц остается неизменной, то есть улучшения, найденные частицами, не влияют на их дальнейшие перемещения. Таким образом изучается только одна частица, а не весь рой. Стохастичность устраняется путем использования математических ожиданий, так что учитываются усредненные и детерминированные поведения частицы. В результате из анализа в значительной мере устраняются все наиболее важные особенности алгоритмов роевого интеллекта.

Другим подходом, часто применяемым для улучшения роевых алгоритмов, является разработка новых модификаций. Эберхарт и его соавторы используют

модификацию, в которой инерция линейно уменьшается в процессе оптимизации [Eberhart, 2000]. В работе [Fan, 2002] введен вспомогательный алгоритм, который сужает границы скоростей частиц на каждой итерации с целью сфокусировать поиск ближе к концу процесса оптимизации. Ван Бергом было разработано несколько вариантов модификаций [Bergh, 2001] исходя из анализа сходимости частиц [Carlisle, 2001; Pedersen, 2010a].

Существует огромное количество модификаций роевых алгоритмов, в первую очередь, алгоритма роя частиц и колонии муравьев. Но ни одна из модификаций не стала популярнее классических версий. Каждая модификация несет в себе дополнительное эвристическое усложнение алгоритма, увеличивая его степень свободы (число коэффициентов, число комбинаций дополнительных правил и т.д.). В результате задача настройки роевых алгоритмов, их понимания, изучения, выбора только усложняется.

В настоящее время настройка роевых алгоритмов выполняется либо вручную, что занимает больше времени, чем само решение задачи (М. Педерсен называет такой способ традиционным, самым популярным и самым неэффективным [Pedersen, 2010a]), либо путем введения дополнительных правил, которые в свою очередь тоже требуют тщательной настройки, либо с помощью различных мета-оптимизационных алгоритмов, управляющих подстройкой в автоматическом режиме, что опять же усложняет алгоритм и приводит к существенному снижению быстродействия. Кроме того, для различных задач оптимизации лучше подходят различные роевые алгоритмы, но методик их выбора не существует.

В результате указанных сложностей, которые можно обобщенно назвать сложностями адаптации, алгоритмы роевого интеллекта недостаточно эффективно применяются на практике. Методы настройки роевых алгоритмов, послужившие основой для разработанных адаптивных алгоритмов роевого интеллекта, изложены в разделе 2.

## 1.4. Выводы по разделу 1

Проведено исследование существующих методов оптимизации, истории их развития и классификации. Выявлены особенности задач оптимизации в проектировании и управлении техническими системами: сложная топология пространства поиска решений, многокритериальность, многофакторность, наличие как дискретных, так и непрерывных управляемых переменных, и ограничений, динамические изменения условий задач, высокая вычислительная сложность. Показана целесообразность применения алгоритмов роевого интеллекта в решении оптимизационных задач для повышения технико-экономических показателей их функционирования. Данные задачи требуют применения методов, имеющих интеллектуальные свойства самоорганизации, позволяющие находить решения с высоким качеством за приемлемое время или очень быстро находить решения с удовлетворительным качеством. К таким методам относятся алгоритмы роевого интеллекта, эффективность которых основана на механизмах решения задач, созданных природой за миллионы лет и при этом имеющих строгие математические доказательства эффективности, основанные на теории марковских процессов.

Выявлены трудности, препятствующие эффективному применению алгоритмов роевого интеллекта на практике: отсутствие единой классификации и терминологии, отсутствие рекомендаций по выбору роевого алгоритма для тех или иных задач, необходимость доработки алгоритмов под различные оптимизационные задачи, необходимость настройки эвристических параметров алгоритмов. Актуальность работы заключается в систематизации алгоритмов роевого интеллекта, повышении их адаптивных свойств для эффективного применения на практике в задачах проектирования и управления, и, в конечном, счете, улучшения функционирования технических систем.

# 2. Адаптивные алгоритмы роевого интеллекта

Раздел 2 посвящен описанию разработанного в диссертационной работе системного подхода к описанию алгоритмов РИ и повышению их адаптивных свойств. В рассматриваемой области проектирования и управления техническими системами оптимизационные задачи отличаются высоким разнообразием, поэтому и возникает необходимость в эффективном и удобном в применении инструменте их решения. Раздел состоит из четырех подразделов. Подраздел 2.1 описывает системное представление алгоритмов РИ, используемую далее терминологию, схему работы всех роевых алгоритмов и принцип их взаимодействия с моделью оптимизируемого объекта. Подраздел 2.2 содержит полученные на основе разработанных схем описания пяти наиболее распространенных роевых алгоритмов: роя частиц, колонии муравьев, роя светлячков, косяка рыб и роя пчел. Подраздел 2.3 посвящен адаптации роевых алгоритмов под условия решаемых задач, которая включает в себя, во-первых, подстройку для применения одних и тех же алгоритмов для различных типов оптимизационных задач, во-вторых, повышение эффективности работы алгоритмов за счет выбора значений их эвристических параметров. Методики повышения эффективности практического применения алгоритмов РИ описаны в подразделе 2.4.

## 2.1. Системное представление алгоритмов роевого интеллекта

### 2.1.1. Введенные обозначения

Как описано в разделе 1, рой является децентрализованной системой, состоящей из множества простых и однообразных элементов, которые косвенно взаимодействуют между собой и с окружающей средой для достижения определенной цели. В различных описаниях роевых алгоритмов используются различные обозначения, в том числе для понятия элемента. В работе [Reynolds, 1987] используется понятие «boids», однако оно не стало общепринятым, на русском языке не имеет аналогов. Использование термина

«агент» можно встретить во многих работах, например, [Курейчик, 2017; Ilie, 2013]. Однако при использовании этого термина возникает противоречие с более широким понятием агента, используемым в мультиагентных системах и с понятием агента, используемым в значении автономной компьютерной программы. Если в концепции РИ элемент по определению является примитивным по структуре и поведению и все элементы роя единообразны, то в мультиагентных системах имеется большое разнообразие агентов, они могут иметь сложные индивидуальные сценарии работы.

В области управления техническими системами под агентом может пониматься сложный объект. Так в электроэнергетике в задаче оптимизации системы обмена электроэнергией агентом может выступать электростанция, потребитель электроэнергии, даже регион или страна [Рыжов, 2014]. Задача становится все более актуальной с развитием малой генерации, использовании солнечной, ветровой энергии. Для решения таких задач часто применяется мультиагентное моделирование, при этом под агентом понимается отдельный активный (генерирующий) потребитель. Но правила взаимодействия агентов могут быть оптимизированы с помощью алгоритмов роевого интеллекта. В этом случае использование термина «агент» будет нести два смысла – активный потребитель и элемент роя, что неизбежно приведет к двусмысленности и неясностям при описании оптимизационной задачи и ее решения.

Существует альтернативный путь – использование слов, взятых из природной основы алгоритма: «пчела», «муравей», «светлячок» и т.п. Данный пусть не позволяет ввести единую схему описания алгоритмов РИ. Кроме того, в случае гибридных алгоритмов возникают слишком громоздкие термины, как, например, в работе [Лебедев, 2013] – «агент-пчеломуравей».

Поэтому в настоящей диссертационной работе выполнен отказ от использования термина «агент» или биологических терминов. Вместо этого предлагается использовать термин «частица». У данного подхода есть ряд преимуществ:

- термин «частица» по своей семантике уже подразумевает примитивность и, в меньшей степени, единообразие, в отличие от термина «агент»;

- термин не входит в конфликт с термином «агент», имеющим как сходство, так и принципиальные отличия в других областях;

- термин является осмысленным, в отличие от «boid» (непереводимого к тому же на русский язык);

- термин «частица» имеет точный аналог на английском языке – «particle»;

- термин не привязан ни к одному из алгоритмов роевого интеллекта, кроме алгоритма роя частиц.

Последний пункт необходимо пояснить дополнительно. Алгоритм роя частиц можно считать одним из двух первых широко распространившихся роевых алгоритмов, наравне с алгоритмом колонии муравьев. И он основан на моделировании поведения птиц [Kennedy, 1995], однако авторы предусмотрительно заменили слово «птица» («bird») на слово «частица» («particle»). Впоследствии авторы новых алгоритмов РИ использовали каждый свой собственный термин, что вносило запутанность в общую концепцию. Поэтому в данной работе термин «частица» выбран как единый для всех алгоритмов РИ.

Кроме того, в русскоязычной литературе существует сложность в разделении терминов «алгоритм» и «метод». Принято использовать термины «метод роя частиц», но «генетический алгоритм» (он не относится к роевым, это пример, показывающий, что сущности одного уровня абстракции именуются разными терминами); «муравьиный алгоритм» и, одновременно, «метод колонии муравьев». В данной работе предлагается называть определенные роевые алгоритмы алгоритмами.

В работе введены следующие обозначения.

*f(X)* – критерий оптимальности или целевая функция, для которой требуется найти наилучшее (минимальное или максимальное) значение, далее, если это не оговорено особо, будет подразумеваться минимум;

$$f^{opt} = f(X^{ot}) = \arg\min_{X \in D} f(X).$$

$X$ – вектор управляемых переменных или варьируемых параметров;

$D$ – область допустимых значений $X$, $D \subset R^{|\mathbf{X}|}$, пространство поиска решений;

$|S|$ – количество частиц роя;

$S = \{s_1, s_2, \ldots, s_{|S|}\}$ – множество всех частиц роя;

$X_{ij}$ – вектор управляемых переменных $i$-й частицы на $j$-й итерации алгоритма, иными словами, положение частицы, ее позиция.

$X_{ij}^{best}$ – наилучшее положение $i$-й частицы от первой до $j$-й итерации;

$X^{opt}$ – оптимальное решение задачи;

$f^{opt}$ – наилучшее значение критерия;

$X_j^{best}$ – наилучшее решение, полученное алгоритмом от первой до $j$-й итерации алгоритма;

$X_{final}^{best}$ – наилучшее значение вектора варьируемых параметров к моменту завершения алгоритма;

$Rnd(a; b)$ – вектор случайных чисел, равномерно распределенных на интервале от $a$ до $b$.

### 2.1.2. Роевой алгоритм как система

Для рассмотрения алгоритмов РИ с позиций системного анализа следует провести соответствия между понятиями системного анализа и РИ. Использованы понятия системного анализа из [Красов, 2000]. Представленные ниже описания впервые опубликованы в работах автора диссертации [Матренин, 2013а; 2015; 2016]. В настоящей работе описания пересмотрены и несколько усовершенствованы, в частности, исключен термин «агент».

1. «Система есть нечто целое». Очевидно, что рой, разделенный на отдельные частицы, не сможет эффективно решать задачи оптимизации.

2. «Система есть организованное множество». Рой является множеством частиц, взаимодействующим по некоторым правилам, что приводит к самоорганизации.

3. «Система есть множество вещей, свойств и отношений». Под вещами можно понимать частицы, под свойствами их характеристики, положения, под отношениями – алгоритм работы роя.

4. «Система есть множество элементов, образующих структуру и обеспечивающих определенное поведение в условиях окружающей среды». По сути, отличие этого определения лишь во введении понятия окружающей среды. Для роя окружающей средой является критерий оптимальности и пространство поиска решений.

5. «Система есть множество входов, множество выходов, множество состояний, характеризуемых операторами переходов и выходов». На вход роя подается задача оптимизации, состояния роя – это состояния всех его частиц на *j*-й итерации, операторы переходов задаются алгоритмом работы роя, выходом является наилучшее найденное значение вектора варьируемых параметров $X_{final}^{best}$.

Выделим общие для всех алгоритмов роевого интеллекта понятия на основании системного описания, используя термины из [Красов, 2000].

1. Элемент. «Под элементом принято понимать простейшую неделимую часть системы». Элементом роя является частица.

2. Подсистема. «Система может быть разделена на элементы не сразу, а последовательным расчленением на подсистемы, которые представляют собой компоненты более крупные, чем элементы, и в то же время более детальные, чем система в целом». Концепция роевого интеллекта допускает выделение групп частиц. Например, возможны следующие ситуации:

- связи между частиц в группе сильнее, чем между частицами из различных групп (островковая модификация алгоритма роя частиц);

- выделяется группа частиц с особым поведением, например, не использующих информацию о предыдущем опыте, для предотвращения преждевременной сходимости алгоритма (разведчики в методе роя пчел).

3. «Структура – это совокупность элементов и связей между ними». Структура роевых систем задается правилами поведения частиц и обменом

информацией. Особенностью таких систем является отсутствие центра, который управлял бы частицами напрямую.

4. Связь между частицами происходит путем косвенного обмена опытом.

5. Состояние. «Понятием "состояние" обычно характеризуют мгновенную фотографию, "срез" системы». Под состоянием понимается состояния всех частиц, состояние объекта для косвенного обмена опытом (наилучшее значение $X_j^{best}$, веса дуг графа с феромоном и т.д.), текущее значение наилучшего вектора варьируемых параметров и наилучшее найденное значение критерия.

6. Поведение. «Если система способна переходить из одного состояния в другое, то говорят, что она обладает поведением» [Красов, 2000]. Поведение роя задается рассмотренным алгоритмом работы роя, который переводит множество частиц из одного состояния в другое.

7. Внешняя среда. «Под внешней средой понимается множество элементов, которые не входят в систему, но изменение их состояния вызывает изменение поведения системы». В данном случае средой является решаемая задача, то есть критерий и ограничения.

8. Равновесие, устойчивость и развитие. С точки зрения роевого интеллекта, эти понятия связаны со сходимостью алгоритмов. Сходимость алгоритмов роевого интеллекта является отдельной областью исследований. Алгоритмы роевого интеллекта должны обеспечивать развитие от начального случайного набора решений до оптимального или близкого к оптимальному решению. При этом высокая устойчивость может привести к быстрой сходимости поиска как к оптимальному решению, так и к некоторому локальному экстремуму, далекому от оптимального. Низкая устойчивость приводит к повышению вероятности попасть в область глобального экстремума, но при этом повышается вероятность и преждевременного выхода из этой области из-за недостаточно подробного ее исследования.

9. Целью роя является нахождение такого значения вектора варьируемых параметров $X$, который обеспечил бы наилучшее значение критерия $f(X)$. Достижение цели не гарантируется, но эффективность стохастической

оптимизации связана с нахождение близких к оптимальным решениям за короткое время в тех задачах, где другие методы неприменимы или малоэффективны.

Подводя итоги проведенного анализа, можно кратко сформулировать следующее. РИ использует множество частиц как элементы; правила их перемещения как правила взаимодействия элементов системы. Отличительной особенностью РИ является использование средства косвенного обмена информацией между частицами. Поскольку алгоритмы РИ являются эвристическими, их работа зависит от значений эвристических параметров. Наконец, алгоритм РИ имеет некие информационные каналы взаимодействия с надсистемой. Из данного описания можно выделить шесть ключевых понятий для понимания и описания роевых алгоритмов.

1. Множество частиц *S*.
2. Средство косвенного обмена информацией между частицами *M*.
3. Алгоритм перемещения частиц *A*.
4. Эвристические параметры *P*.
5. Вход для получения данных извне *I*.
6. Выход для отправки данных вовне *O*.

### 2.1.3. Обобщенная схема работы роевого алгоритма

Работа алгоритма РИ включает в себя следующие шаги.

1. Генерация начального множества частиц. Некоторым образом, как правило, с использование рандомизации, в пространстве поиска распределяются частицы. Номер итерации $j=1$.

2. Вычисление критерия для каждой частицы как функции от позиции частицы $f(X_{ij})$. Для отдельных алгоритмов, таких как, алгорим колонии муравьев, вычисление критерия выполняется после перемещения частиц, поэтому для них второй шаг на первой итерации пропускается.

3. Перемещение частиц (миграция). На этом шаге реализуется главная особенность алгоритмов роевого интеллекта – выполнение каждой частицей своих действий на основании:

- индивидуальных правил и опыта;
- косвенного обмена информацией с другими частицами роя;
- стохастических свойств.

4. Проверка условия завершения. Если условие выполнено, процесс завершается, значение $X_{final}^{best}$ будет конечным результатом, иначе происходит переход к шагу 2 (с увеличением номера итерации $j$ на единицу).

Условие завершения могут быть связано с выполнением заранее заданного числа итераций, или обнаружением решения не хуже указанного как удовлетворительного, или стагнацией поиска ($X_j^{best}$ не меняется в течение определенного числа итераций алгоритма).

Анализ третьего шага алгоритма позволяет определить, относится ли он к алгоритмам роевого интеллекта. В роевых алгоритмах, в формулах, определяющих поведение частиц, есть либо элемент, связанный с одним или несколькими наилучшими положениями, найденными всем роем, либо средневзвешенное позиция частиц роя (центр тяжести), где веса пропорциональны значению критерию, соответствующего позиции частицы. Так, в алгоритме роя частиц для организации обмена информацией между частицами используется наилучшее положение среди всех частиц и всех выполненных итераций $X_j^{best}$, в алгоритме роя пчел – несколько наилучших позиций частиц, в обезьяньем поиске – центр тяжести. В алгоритме муравьиной колонии для взаимодействия используется граф, веса дуг которого изменяются в зависимости от того, какое значение критерия было получено при движении частиц по нему.

В отличии от роевого принципа обмена информацией, в эволюционных алгоритмах отсутствует косвенное взаимодействие, вместо него используется процесс отбора решений с наилучшим значением критерия.

### 2.1.4. Взаимодействие роевого алгоритма и решаемой задачи

Для взаимодействия алгоритма РИ и модели оптимизируемого объекта, предложен следующий интерфейс и алгоритм взаимодействия (рисунок 2.1). Под моделью оптимизируемого объекта понимается математическая или программная модель, на которой можно выполнить расчеты с целью определения эффективности различных решений (альтернатив). Моделью может выступать и непосредственно сам объект, например, в задачах оперативного управления.

1. Алгоритм РИ получает размерность пространства поиска решения через вход $I_{dim}$. Эта величина определяет длину вектора $X$.

2. Алгоритм РИ генерирует варианты решений задачи путем перемещения частиц. Каждая частица представляет один вариант решения задачи. Подраздел 2.3.1 показывает, как именно позиция частицы может быть переведена в решение оптимизационной задачи.

3. Все варианты решений отправляются в модель оптимизируемого объекта через выход $O_{var}$.

4. Полученные варианты решения проверяются на модели оптимизируемого объекта для определения допустимости с точки зрения ограничений и определения значения критерия (критериев).

5. Результаты моделирования возвращаются в алгоритм РИ через вход $I_{crit}$.

6. Полученные результаты влияют на перемещения частиц, таким образом, происходит переход к шагу 2. Либо, если выполнено условие завершения алгоритма, то результат его работы (наилучшее найденное решение или множество альтернатив) передается в надсистему через выход $O_{best}$.

На рисунке 2.1 утолщением выделены линии, показывающие многократно выполняемый обмен вариантами решений от алгоритма к модели и – в обратную сторону – результатов оценки решений.

Приведенный алгоритм взаимодействия показывает, что модель оптимизируемого объекта и метод расчета критерия не влияют на роевой алгоритм. Это обеспечивает независимость алгоритма и оптимизационной задачи. Таким

образом, алгоритм роевого интеллекта и модель оптимизируемого объекта представляют друг для друга черные ящики.

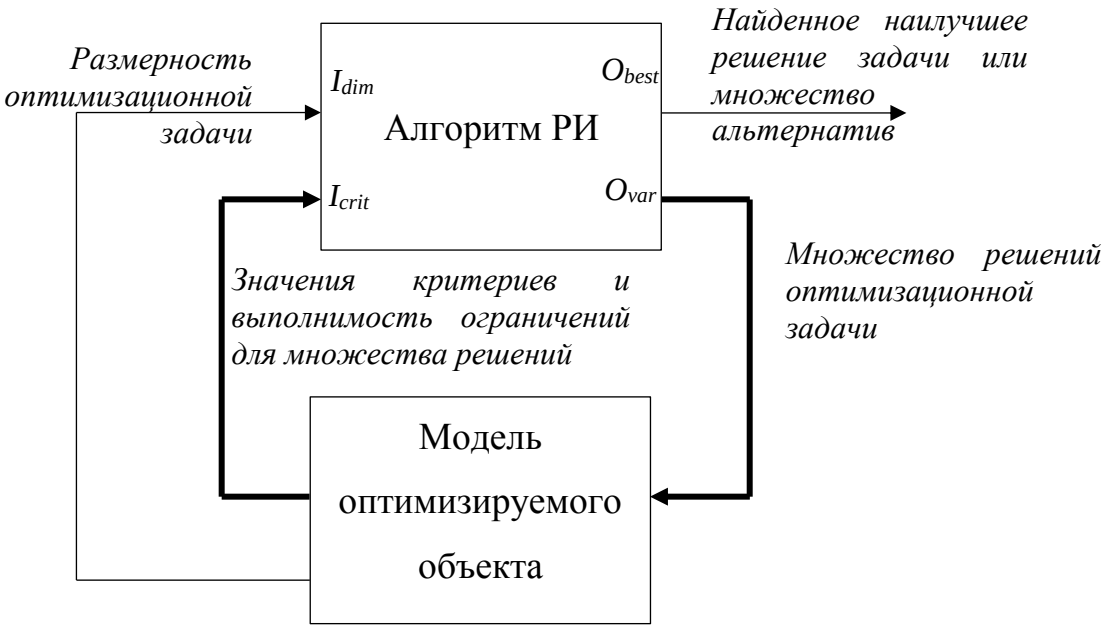



Рисунок 2.1 – Схема взаимодействия алгоритма роевого интеллекта и модели оптимизируемого объекта

Предложенный подход позволяет быстро применять различные роевые алгоритмы к различным оптимизационном задачам, так как не требует внесения изменений ни в алгоритме, ни в модель оптимизируемого объекта. Отсутствие тесных взаимосвязей между задачами и алгоритмами увеличивает гибкость алгоритмов РИ.

## 2.2. Описания роевых алгоритмов по разработанной схеме

### 2.2.1. Алгоритм роя частиц

Алгоритм роя частиц (PSO – Particle Swarm Optimization) был изначально разработан для визуализации движения стай птиц в компьютерной графике [Reynolds, 1987]. В 1995 году Дж. Кеннеди и Р. Эберхарт предложили этот алгоритм для решения задач непрерывной глобальной оптимизации [Eberhart, 1995]. Затем алгоритм был доработан указанными авторами и О. Ши [Kennedy, 1995; Shi, 1998a]

до той версии, которая в настоящее время считается классической. Как отмечалось в разделе 1, существует множество модификаций алгоритма, но ни одна из них не стала на данный момент общепринятой, как классический вариант.

Стая птиц действует скоординировано, так как одной птице сложно быстро найти источники пищи, только если птицы будут обмениваться информацией, вся стая сможет выжить. В алгоритме в качестве средства обмена данными используется так называемая общая память, суть которой в том, что, наилучшее решение, найденное роем, известно всем его частицам.

Согласно схеме описания алгоритмов РИ (подраздел 2.1), алгоритм роя частиц можно записать в виде $PSO = \{S, M, A, P, I, O\}$.

**Множество частиц** $\boldsymbol{S} = \{s_1, s_2, \ldots, s_{|S|}\}$, $|S|$ – количество частиц. На $j$-й итерации $i$-я частица характеризуется состоянием $s_{ij} = \{X_{ij}, V_{ij}, X^{best}_{ij}\}$, где $X_{ij} = \{x^1_{ij}, x^2_{ij}, \ldots, x^l_{ij},\}$ – вектор варьируемых параметров (положение частицы), $V_{ij} = \{v^1_{ij}, v^2_{ij}, \ldots, v^l_{ij}\}$ – вектор скоростей частицы, $X^{best}_{ij} = \{b^1_{ij}, b^2_{ij}, \ldots, b^l_{ij}\}$ – наилучше по значению фитнесс-функции положение частицы среди всех положений, которые она занимала в процессе работы алгоритма от 1-й до $j$-й итераций, $l$ – количество варьируемых параметров.

**Средство косвенного обмена информацией между частицами** $\boldsymbol{M} = X_j^{best}$ – наилучшее значение вектора варьируемых параметров, которое было получено среди всех частиц от 1-й до $j$-й итерации алгоритма. Этот вектор обеспечивает косвенный обмен опытом между частицами.

**Алгоритм** $\boldsymbol{A}$ описывает перемещение частиц. Далее представлено описание базового алгоритма с некоторыми дополнениями, введенными в данной работе.

1. Генерация начального множества частиц ($j$=1):

$$X_{i1} = Rnd(0;1), i = 1,..,|S|\ ,$$

$$V_{i1} = Rnd(-\beta;\beta), i = 1,..,|S|\ ,$$

в данной работе вводится настраиваемое ограничение на максимальную скорость частиц.

$$X^{best}_{i1} = X_{ij}, i = 1,..,|S|\ .$$

Произвольно выбирается наилучшая позиция (при вычислении фитнесс-функций будет определена действительно наилучшая позиция):

$$X_1^{best} = X_{11} .$$

2. Вычисление фитнесс-функций, определение наилучшего индивидуального положения частиц. Обновление наилучшего найденного решения $X_1^{best}$.

$$X_{ij}^{best} = X_{ij}, f(X_{ij}) < f(X_{ij}^{best}) , i = 1,..,|S| ,$$

Вычисление $f(X)$ происходит во внешней среде, с помощью обмена данными по обратной связи ($O_{var}$, $I_{crit}$) (рисунок 2.1).

3. Перемещения частиц:

$$V_{ij+1} = V_{ij}\omega + \alpha_1\left(X_{ij}^{best} - X_{ij}\right)Rnd(0;1)$$

$$+\alpha_2\left(M - X_{ij}\right)Rnd(0;1), i = 1,..,|S| \quad , (2.1)$$

$$V_{ij+1} = \text{clamp}(V_{ij+1}; -\beta; \beta), i = 1,..,|S| ,$$

$$X_{ij+1} = \text{clamp}(X_{ij} + V_{ij+1}; 0; 1), i = 1,..,|S| ,$$

4. Если на $j$-ой итерации выполнено условие остановки, то значение $X_{final}^{best} = X_j^{best}$ подается на выход $O_{best}$. Иначе происходит переход к итерации 2.

**Вектор *P*** = {$\alpha_1$, $\alpha_2$, $\omega$, $\beta$} – коэффициенты алгоритма, которые влияют на процесс движения частиц в пространстве поиска решений. Коэффициенты $\alpha_1$ и $\alpha_2$ определяют, соответственно, степень учета индивидуального и группового опыта частиц. Коэффициент $\omega$ характеризует инерционные свойства частиц.

В данной работе предложено ввести ограничение на максимальную скорость частиц, коэффициент $\beta$. В оригинальном алгоритме роя частиц [Eberhart, 1995], а также работах по изучению влияния коэффициентов на работу алгоритма [Pedersen, 2010a; Pedersen, 2010b; Shi, 1998b] скорость частиц ограничивается только размером пространства поиска решений, так что за один шаг частица может переместиться от минимального значения координаты до максимального. Однако, когда частицы могут перемещаться на большие расстояния, повышается

вероятность пропуска более эффективных решений задачи. Введение дополнительного ограничения на скорость частиц снижает риск пропуска окрестности глобального экстремума, при этом используемый в данной работе подход не усложнят алгоритм, в отличие от существующих методов по динамическому регулированию скоростей, таких как [Fan, 2002]. Но на более поздних итерациях алгоритма частицы, как правило, и не разгоняются до высоких скоростей, так как разности ($X_{ij}^{best} - X_{ij}$) и ($M - X_{ij}$) в выражении (2.1) уменьшаются, когда частицы роя группируются в окрестности найденного решения. Кроме того, этот метод не помогает избегать пропуска эффективных оптимумов на начальных итерациях алгоритма, пока скорости частиц высоки. Если же коэффициент β равен единице, то предложенная модификация вырождается в классический алгоритм роя частиц.

### 2.2.2. Алгоритм колонии муравьев

Алгоритм колонии муравьев (Ant Colony Optimization – ACO) предложен в начале 90-х годов году бельгийским ученым Марко Дориго [Dorigo, 1996; 2006]. Муравьи решают задачу поиска кратчайших путей с помощью коллективного взаимодействия на базе химической регуляции [Dorigo, 1996; Штовба, 2003]. Муравей при своем движении оставляет особые вещества – феромоны. Другие муравьи реагируют на эти вещества, так что чем больше по некоторому пути прошло муравьев, тем сильнее запах на нем, и тем выше вероятность, что муравей будет двигаться по тому же пути. Муравьи, нашедшие путь, отнимающий меньше времени на прохождение от муравейника до источника пищи, успевают пройти по нему чаще, поэтому их маршрут становится более заметным и привлекает все большее количество муравьев. Менее используемые пути постепенно теряют запах, то есть пропадают. Алгоритм сводится к обходу взвешенного графа, дуги которого имеют параметр, называемый феромоном или количеством феромона. Таким образом в основе модели лежит косвенный обмен информацией между частицами через особую среду, представленную взвешенным графом. Для решения задач оптимизации алгоритмом колонии муравьев необходимо представить задачу

в виде нахождения кратчайшего маршрута на взвешенном графе. Классический муравьиный алгоритм направлен на решение комбинаторных задач, таких как задача коммивояжера и ее производных. Поэтому вектором управляемых переменных является последовательность узлов графа как оптимальный по некоторому критерию маршрут.

Согласно схеме описания алгоритмов РИ (подраздел 2.1), алгоритм роя частиц можно записать в виде $ACO = \{S, M, A, P, I, O\}$.

**Множество частиц (муравьев) $S$** = $\{s_1, s_2, \ldots, s_{|S|}\}$, $|S|$ – количество частиц. На $j$-й итерации $i$-я частица характеризуется состоянием $s_{ij} = \{X_{ij}, T_{ij},\}$, где $X_{ij} = \{x^1_{ij}, x^2_{ij}, \ldots, x^l_{ij}\}$ – вектор варьируемых параметров (последовательность узлов графа), $T_{ij} = \{t^1_{ij}, t^2_{ij}, \ldots, t^l_{ij}\}$ – вектор булевых переменных, которые показывают, был ли $l$-й узел посещен $i$-й частицей на $j$-й итерации (в начале каждой итерации все его компоненты равны 0).

**Средством косвенного обмена информацией между частицами служит граф $M$** = $\{F, R\}$, каждая дуга которого имеет два веса: переменный и постоянный. Постоянной составляющей может и не быть. В задачах поиска кратчайших путей, постоянная составляющая, как правило характеризует расстояние между пунктами. Поэтому граф $M$ можно представить в виде двух графов с одинаковыми структурами, но разными весами дуг ($F$ и $R$):

$$F = \begin{pmatrix} \tau_{11} & \cdots & \tau_{1l} \\ \vdots & \ddots & \vdots \\ \tau_{l1} & \cdots & \tau_{ll} \end{pmatrix}, R = \begin{pmatrix} r_{11} & \cdots & r_{1l} \\ \vdots & \ddots & \vdots \\ r_{l1} & \cdots & r_{ll} \end{pmatrix},$$

где $\tau_{k1k2}$ – количество феромона на дуге, соединяющей $k_1$-й и $k_2$-й узлы графа $F$, $r_{k1k2}$ – вес (длина) дуги, соединяющей $k_1$-й и $k_2$-й узлы графа $R$, при этом в общем случае $\tau_{k1k2} \neq \tau_{k2k1}$ и $r_{k1k2} \neq r_{k2k1}$.

**Алгоритм $A$** описывает перемещение частиц.

1. Генерация начального множества частиц. В зависимости от задачи, каждая частица может быть создана в случайном узле графа или в заданном. Если частица помещена в узел $k$, то

$$x^1_{i1}=k,\ t^k_{i1}=1.$$

На каждую дугу наносится некоторое ненулевое количество феромона

$$\tau_{ij} = \tau_{min}.$$

2. Перемещения частиц. В алгоритме колонии муравьев на первой итерации ($j = 1$) второй шаг пропускается, так как для вычисления критериев необходимо выполнить перемещения частиц.

Для прочих шагов выполняются действия:

- вычисление критериев оптимальности: $f(X_{ij}), i = 1,..,|S|$, Обновление наилучшего найденного решения $X_1^{best}$.

- определение количества феромона, которое нужно нанести на дугу, соединяющую узлы $k_1$ и $k_2$ (феромон для $i$-й частицы наносится только на те дуги, которые вошли в маршрут $X_{ij}$):

$$\Delta\tau_{ij}^{k_1k_2} = \begin{cases} \left(\frac{\gamma}{f(X_{ij})}\right)^{\mu}, \text{в } X_{ij}\ k_1 \text{ следует сразу за } k_2 \\ 0, \text{иначе} \end{cases} \quad (2.2)$$

$$\tau_{ij+1}^{k_1k_2} = \tau_{ij}^{k_1k_2} + \Delta\tau_{ij}^{k_1k_2}$$

$$, i = 1,..,|S| \text{ , } k_1\text{=1,.., } l \text{ , } k_1\text{=1,.., } l$$

- усиление феромона на наилучшем известном пути:

$$\tau_{bj+1}^{k_1k_2} = \min(\tau_{bj+1}^{k_1k_2} \cdot \lambda, \tau_{max})$$

- пересчет количества феромона на всем графе с учетом испарения и ограничений:

$$\theta = \rho \cdot \tau_j^{k_1k_2} + \Delta\tau_{ij}^{k_1k_2}, k_1 = 1,..,l, k_2 = 1,..,l\text{ ,}$$

$$\tau_j^{k_1k_2} = \begin{cases} \theta, \theta \geq \tau_{min} \\ \tau_{min}, \theta < \tau_{min} \end{cases}.$$

3. Перемещения частиц. В каждом узле $k$ частица выбирает, в какой из еще не посещенных узлов перейти. При этом вероятность перехода в $m$-й узел равна (индексы $i$ и $j$, определяющие частицу и итерацию, здесь опущены):

$$\Phi_m = \begin{cases} \frac{(\tau_{km})^{\alpha} \cdot (\eta(r_{km}))^{\beta}}{\sum_{z=1,t_z=0}^{l}((\tau_{kz})^{\alpha} \cdot (\eta(r_{kz}))^{\beta})}, t_m = 0 \\ 0, t_m = 1 \end{cases}. \quad (2.3)$$

Здесь $\eta(r_{km})$ – некоторая функция от веса дуги, в простейшем случае

$$\eta(r_{km}) = r_{km}\ .$$

После вычисления вероятностей с помощью розыгрыша по жребию происходит определение, в какой узел $m$ должна переместиться частица. При этом $t_m = 1$, номер узла добавляется в маршрут частицы. После окончания обхода вектор посещенных узлов $T$ обнуляется.

4. Если на $j$-ой итерации выполнено условие остановки, то значение $X_{final}^{best} = X_j^{best}$ подается на выход $O_{best}$. Иначе происходит переход к итерации 2.

**Вектор** $\boldsymbol{P} = \{\alpha, \beta, \gamma, \lambda, \rho\}$ – коэффициенты алгоритма. Коэффициент α влияет на зависимость между количеством феромона на дуге и вероятностью ее выбора частицей. Коэффициент β определяет степень влияния веса дуги на вероятность ее выбора. Коэффициент γ – коэффициент интенсивности выделения феромона. Коэффициент λ способствует привлечению большего количества частиц к наилучшему найденному решению, как более перспективному. Коэффициент μ обеспечивает нелинейность зависимости феромона, наносимого на дуги маршрута от значения критерия, полученного при обходе графа по данному маршруту. Коэффициент ρ определяет скорость испаряемости феромона.

### 2.2.3. Алгоритм роя светлячков

Алгоритм светлячков (FireFly), как и все алгоритмы роевого интеллекта, основан на перемещении частиц (светлячков) в пространстве поиска решений. Был предложен в 2008 году Ш. Янгом [Yang, 2010b]. Поскольку алгоритм основан на наблюдении за поведением светлячков, то считается, что каждая частица видит «свет» от своих соседей, но яркость «света» зависит от расстояния между частицами. Частица учитывает только соседей с лучшим, чем у нее, значением критерия. Кроме того, как и во всех роевых алгоритмах, в перемещениях частиц используется стохастичность.

Согласно схеме описания алгоритмов РИ (подраздел 2.1), алгоритм светляков можно представить в виде $FFO = \{S, M, A, P, I, O\}$.

**Множество частиц (светлячков) $S$** = $\{s_1, s_2,\ldots,s_{|S|}\}$, $|S|$ – количество частиц. На $j$-й итерации $i$-я частица характеризуется состоянием $s_{ij} = \{X_{ij}\}$, где $X_{ij}=\{x^1_{ij}, x^2_{ij},\ldots, x^l_{ij}\}$ – вектор варьируемых параметров (положение частицы), $l$ – количество варьируемых параметров.

**Средством косвенного обмена информацией между частицами служит вектор $M$** = $\{f(X_{1j}), f(X_{2j}),\ldots, f(X_{|s|j})\}$ – яркости свечений частиц. Свечение определяется критерием оптимальности. Этот вектор обеспечивает косвенный обмен опытом между частицами.

**Алгоритм $A$** определяет перемещение частиц.

1. Генерация начального множества частиц ($j$=1):

$$X_{i1} = Rnd(0;1), i = 1,..,|S| .$$

Произвольно выбирается наилучшая позиция:

$$X_1^{best} = X_{11} .$$

2. Вычисление критерия по каждой частице. Обновление наилучшего найденного решения $X_1^{best}$.

$$m_{ij} = f(X_{ij}), i = 1,\ldots,|S|$$

Вычисление $f(X)$ происходит в математической модели задачи, куда из алгоритма поступают векторы $X_{ij}$ , а результаты возвращаются в алгоритм

3. Перемещения частиц.

$$X_{ij+1} = X_{ij} + v(X_{ij}, X_{kj}) * (X_{kj} - X_{ij}) + \alpha * Rnd(-1;1) \ ,$$

$$i,k = 1,\ldots,|S|, i \neq k,$$

$$X_{ij+1} = \text{clamp}(X_{ij} + V_{ij+1}; 0; 1), i = 1,..,|S|$$

Функция $v(X_{ij}, X_{kj})$ определяет притягательность $k$-ой частицы для $j$-й частицы на $j$-й итерации алгоритма:

$$v(X_{ij}, X_{kj}) = \begin{cases} \dfrac{\beta}{1+\gamma r(X_{ij}, X_{kj})}, m_{kj} < m_{ij} \\ 0,\ f(X_{kj}) \geq f(X_{ij}) \end{cases}, i = 1,\ldots,|S|$$

где $r(X_{ij}, X_{kj})$ – декартово расстояние между частицами.

4. Если на *j*-ой итерации выполнено условие остановки, то значение $X_j^{best}$ подается на выход *O*. Иначе происходит переход к итерации 3.2.

**Вектор *P*** = {α, β, γ} – коэффициенты алгоритма. Коэффициент α влияет на уровень недетерминированности алгоритма, чем он ниже, тем движение частиц более определено и предсказуемо. Коэффициенты β и γ определяют степень взаимного влияния частиц в зависимости от расстояния между ними. При нулевом значении β алгоритм вырождается в случайный поиск. При нулевом значении γ расстояние не влияет на притяжение частиц, а чем выше коэффициент γ, тем быстрее убывает степень притяжения частиц с ростом расстояния между ними.

### 2.2.4. Алгоритм косяка рыб

Алгоритм косяка рыб (Fish School Search, FSS) создан в 2008 г. Б. Фило и Л. Нето на основании изучения движения рыб в косяке.

Каждая частица выполняет несколько видов перемещений:

- перемещения на основании индивидуального опыта;
- перемещения на основании опыта всего роя.

Перемещения на основании опыта роя состоят из двух фаз. В отличие от алгоритма роя частиц, при перемещениях напрямую учитываются результаты лишь прошлой итерации. Важно, что для данного алгоритма критерий должен быть неотрицателен на всем пространстве поиска: $f(X) \geq 0$, $X \in D$.

Согласно схеме описания алгоритмов РИ (подраздел 2.1), для получения обобщенного описания, алгоритм поиска косяком рыб необходимо представить в виде

$$FSS = \{S, M, A, P, I, O\}.$$

**Множество частиц (рыб) *S*** = $\{s_1, s_2, \ldots, s_{|S|}\}$, $|S|$ – количество частиц. На *j*-й итерации *i*-ая частица характеризуется состоянием $s_{ij} = \{X_{ij}, V_{ij}, w_{ij}\}$, где $X_{ij} = \{x^1_{ij}, x^2_{ij}, \ldots, x^l_{ij}\}$ – вектор варьируемых параметров (положение частицы), $l$ – размерность пространства поиска решений, $V_{ij} = \{v^1_{ij}, v^2_{ij}, \ldots, v^l_{ij}\}$ – вектор скоростей частицы, $w_{ij}$ – вес *i*- й частицы на *j*-й итерации.

**Средством косвенного обмена *M*** является вектор из двух элементов. Первый из них является скаляром и определяет взвешенную сумму индивидуальных перемещений частиц, а второй является вектором длиной $l$ и представляет взвешенный цент тяжести всего косяка, $M = \{m^S_j, C_j\}$.

**Алгоритм *A*** определяет перемещение частиц.

1. Генерация начального множества частиц ($j$=1):

$$X_{i1} = Rnd(0,1), i = 1,..,|S|$$

$$w_{i1} = \frac{w_{max}}{2}, i = 1,\dots,|S|$$

где $w_{max}$ – один из параметров алгоритма.

2. Вычисление фитнесс-функций каждой частицы на текущей $j$-й итерации: $f(X_{ij})\ i = 1,\dots,|S|$. Обновление наилучшего найденного решения $X_1^{best}$.

3. Перемещения частиц выполняются в нескольких стадий, поэтому вводится промежуточная итерация $j+0{,}5$. Она служит для упрощения дальнейших формул.

3.1. Индивидуальные перемещения частиц состоят, в свою очередь, из трех шагов. На первом шаге генерируется случайное значение скорости частицы:

$$V_{ij+0,5} = Rnd(-1;1)v_{max}, i = 1,\dots,|S|.$$

На втором шаге выполняется перемещение с данной скоростью с учетом области допустимых решений:

$$X_{ij+0,5} = \text{clamp}(X_{ij} + V_{ij+0,5}; 0;\ 1), i = 1,..,|S|$$

На третьем шаге частица возвращается на предыдущую позицию, если значение критерия в новой позиции хуже:

$$X_{ij+0,5} = \begin{cases} X_{ij+0,5}, f(X_{ij+0,5}) \geq f(X_{ij}) \\ X_{ij}, f(X_{ij+0,5}) < f(X_{ij}) \end{cases}, i = 1,\dots,|S|.$$

3.3.2. Инстинктивно-коллективное перемещение выполняется частицами вместе с одинаковой по величине и направлению скоростью. Используется объект для косвенного взаимодействия $m^S_j$:

$$m_j^S = \frac{\sum_i (V_{ij+0,5}(f(X_{ij+0,5}) - f(X_{ij})))}{\sum_i (f(X_{ij+0,5}) - f(X_{ij})))}$$

$$X_{ij+0,5} = X_{ij+0,5} + m_j^S, i = 1,..,|S|$$

$$X_{ij+0,5} = \text{clamp}(X_{ij+0,5}; 0;\ 1), i = 1,..,|S|$$

3.3.3. Коллективно-волевое перемещение тоже выполняется с учетом позиций всех частиц, но каждая совершает его по-своему. Предварительно определяются веса частиц:

$$w_{ij+0,5} = clamp(w_{ij} + \frac{f(X_{ij+0,5}) - f(X_{ij})}{\max\left(f(X_{ij+0,5}), f(X_{ij})\right)}; 1; \leq w_{max})\ \ i = 1,..,|S|,$$

Если после перемещений качество позиций роя в среднем улучшилось, то происходит уменьшение (сужение) текущей области поиска. Иначе область поиска увеличивается, чтобы алгоритм смог выйти из локального экстремума, в который, возможно, попал. Вводится понятие центра тяжести всего роя:

$$C_j = \frac{\sum_i w_{ij+0,5} X_{ij+0,5}}{\sum_i w_{ij+0,5}}\ i = 1, \ldots, |S|.$$

Перемещения при коллективно-волевом плавании выполняется по следующему правилу:

$$X_{ij+1} = \begin{cases} X_{ij+0,5} + vol \cdot (X_{ij+0,5} - C_j), ws_{j+0,5} > ws_{j-0,5} \\ X_{ij+0,5} - vol \cdot (X_{ij+0,5} - C_j), ws_{j+0,5} \leq ws_{j-0,5} \end{cases}, i = 1, \ldots, |S|\ ,$$

где $ws_{j+0,5}$ – сумма весов всех частиц на текущей стадии, $vol$ определяет величину шага перемещений и вычисляется как

$$vol = Rnd(0; 1) \cdot vol_{max},$$

где $vol_{max}$ – максимально возможный размер шага.

Чтобы получить окончательные позиции частиц, нужно учесть границы области допустимых решений:

$$X_{ij+1} = \text{clamp}(X_{ij+1}; 0;\ 1), i = 1,..,|S|$$

**Вектор** $\boldsymbol{P}$ = {$v_{max}$, $vol_{max}$, $w_{max}$} – коэффициенты алгоритма, которые определяют особенности перемещения частиц. Коэффициенты $v_{max}$, $vol_{max}$ ограничивают скорости индивидуального и коллективно-волевого перемещения,

соответственно. Помимо прочего соотношение этих коэффициентов регулирует влияние индивидуального и коллективного поведения частиц в алгоритме. Коэффициент $w_{max}$ ограничивает максимальную значимость отдельной частицы, чтобы алгоритм не был подвержен слишком сильной сходимости в область текущего наилучшего известного решения.

### 2.2.5. Алгоритм роя пчел

Алгоритм роя пчел (Artificial Bee Colony Algorithm или Bees Algorithm) разработан группой авторов и опубликован в 2005 году [Pham, 2005; Karaboga, 2005]. Поведение пчел можно упрощенно описать следующим образом. Рой отправляет пчел-разведчиков в случайных направлениях для поиска нектара. Когда разведчики вернутся, они сообщат рою о найденных участках, и прочие пчелы отправятся на них за нектаром. Причем разведчики перемещаются всегда произвольно, не учитывая предыдущие результаты.

Алгоритм на каждой итерации среди всех частиц выбирает $n^b$ лучших по значению критерия оптимальности, затем среди прочих частиц отбирается $n^g$ лучших, называемых «выбранными» или «перспективными». Задается правило чтобы расстояния между каждой парой позиций в объединенном множестве $n^b \cup n^g$ позиций не превышали заданного порогового значения. Поэтому если две позиции близки, то худшая из них по значению критерия отбрасывается.

Определенные таким образом позиции (множество лучших $N^b$, и множество выбранных $N^g$), запоминаются и на следующем шаге в окрестность каждой лучшей позиции высылается $c^b$ частиц, а каждой выбранной $c^g$. Частица, отправляема в окрестность некоторого участка, оказывается в случайной точке внутри этой окрестности. Так в двумерном пространстве окрестность участка с центром в точке ($x$, $y$) представляет собой область ([$x - rx$; $x+rx$], [$y - ry$; $y+ry$]), где $rx$ и $ry$ – эвристические параметры алгоритма. В настоящей работе, поскольку пространство поиска решений для роевого алгоритма всегда ограничено областью [0, 1], используется единый коэффициент $rx$ по всем измерениям. На каждой итерации

алгоритма частицы-разведчики отправляются в случайные позиции, равномерно распределенные по всему пространству поиска решений. Согласно схеме описания алгоритмов РИ (подраздел 2.1), для получения обобщенного описания алгоритм роя пчел необходимо представить в виде

$$ABCO = \{S, M, A, P, I, O\}.$$

**Множество частиц (пчел) *S*** = $\{s_1, s_2, \ldots, s_{|S|}\}$, $|S|$ – количество частиц. На $j$-й итерации $i$-я частица характеризуется состоянием $s_{ij} = \{X_{ij}\}$, где $X_{ij} = \{x^1_{ij}, x^2_{ij}, \ldots, x^l_{ij}\}$ – вектор варьируемых параметров (положение частицы), $l$ – размерность пространства поиска решений.

**Средством косвенного обмена *M*** является список лучших и перспективных позиций, найденных на $j$-й итерации. Нужно отметить, что в отличие от эволюционных алгоритмов, отбираются не частицы, а только соответствующие позиции в пространстве поиска,

$$M = \{N_{ij}^b, N_{kj}^g\},\ i = 1, \ldots, n^b,\ k = 1, \ldots, n^g.$$

**Алгоритм *A*** определяет перемещение частиц.

1. Генерация начального множества частиц ($j$=1):

$$X_{i1} = Rnd(0;1), i = 1, .., |S|\ ,$$

где $n^s$ – количество частиц-разведчиков.

2. Вычисление фитнесс-функций каждой частицы на текущей $j$-й итерации: $f(X_{ij})\ i = 1, \ldots, |S|$. Обновление наилучшего найденного решения $X_1^{best}$.

3. Перемещения частиц.

3.1. Формирование набора лучших $N_{ij}^b$ и набора перспективных $N_{kj}^g$ найденных на ($j$–1)-й итерации позиций. Среди позиций всех частиц выбирается $N_{ij}^b$ наилучших по значению критерия, среди оставшихся еще $N_{ij}^g$ наилучших. При этом среди выбранных позиций не должно быть близких. Две позиции $X_y$ и $X_z$ считаются близким, если они ближе порога хотя бы по одной координате:

$$\exists l\ \left|x_y^l - x_z^l\right| < rx\ .$$

3.2. Перемещение частиц-рабочих. В окрестность каждой лучшей позиции отправляется $c^b$ частиц, в окрестность каждой перспективной $c^g$ частицы. В некоторых вариантах алгоритма количество отправляющихся в окрестность участка частиц зависит от его качества с точки зрения критерия оптимальности, но в данном описании это не используется. Таким образом, позиции всех частиц-рабочих определяются как:

$$X_{(i-1)c^b+k\ j+1} = N_{ij}^b + Rnd(-1;1)\cdot rad, i = 1, \ldots, n^b, k = 1, \ldots, c^b$$

$$X_{n^b c^b+(i-1)c^b+k\ j+1} = N_{ij}^g + Rnd(-1;1)\cdot rad, i = 1,\ldots, n^g, k = 1, \ldots, c^g\ ,$$

С учетом ограничений:

$$X_{ij+1} = \text{clamp}\left(X_{ij+1}; 0;\ 1\right), i = 1,.., n^b c^b + n^g c^g$$

3.3. Перемещение частиц-разведчиков. Частицы отправляются в случайные позиции, координаты которых равномерно распределены в пространстве допустимых решений:

$$X_{n^b c^b+n^g c^g+i\ j+1} = Rnd(0;1), i = 1, \ldots, n^s.$$

**Коэффициенты (параметры) алгоритма** $P = \{n^s, n^b, n^g, c^b, c^g, rad, rx\}$. Коэффициент *rad* определяет степень рассредоточенности частиц при отправлении на позиции, то есть размеры их окрестностей. Коэффициент *rx* определяет минимальное возможное расстояния между позициями в объединенном множестве лучших и выбранных позиций. Коэффициенты $n^s$, $n^b$, $n^g$, $c^b$, $c^g$ означают количество частиц-разведчиков ($n^s$), частиц, отправляемых на лучшие участки ($n^b$), частиц, отправляемых на выбранные участки ($n^g$), и число лучших и выбранных участков ($c^b$ и $c^g$). Сумма $n^s + n^b c^b + n^g c^g$ равна общему количеству частиц роя |S|.

### 2.2.6. Выделение отличительных черт роевых алгоритмов

Сопоставление данных описаний алгоритмов РИ позволяет выделить принципы, которые использует каждый из алгоритмов, помимо общих принципов РИ (таблица 2.1).

Таблица 2.1 – Индивидуальные особенности роевых алгоритмов

| **Алгоритм** | **Особенности** |
|---|---|
| Роя частиц | - настраиваемое соотношение между важностью индивидуального и коллективного опыта;<br>- использование градиента;<br>- инерция, присущая движению частиц. |
| Колонии муравьев | - позитивная обратная связь;<br>- решение в виде пути на графе;<br>- снижение важности опыта, полученного на прошлых итерациях со временем. |
| Роя светлячков | - каждая частица испытывает влияние всех частиц, занимающих положения лучше, чем она сама;<br>- использование градиента;<br>- влияние частиц друг на друга обратно пропорционально расстоянию между ними. |
| Косяка рыб | - несколько различных фаз на этапе перемещения частиц;<br>- использование градиента;<br>- использование «центра тяжести» частиц. |
| Роя пчел | - часть частиц выполняет полностью случайный поиск, не используя ни свой, ни коллективный опыт;<br>- запрет на сходимость всех частиц в одной области пространства поиска решений. |

Выбирая указанные особенности, можно создавать гибридные алгоритмы РИ, эффективные для решения определенных классов задач. Для задач оптимизации на графах естественным будет выбор алгоритма колонии муравьев, поскольку он по свой природе заключается в поиске некоторого маршрута на графе. Для задач непрерывной нединамической оптимизации, где требуется найти решение как можно лучше и достаточный запас времени, подойдут алгоритмы, в явном виде использующие градиент: роя частиц, роя светлячков, косяка рыб, поскольку их частицы сходятся в окрестность экстремума и затем шаг их перемещений снижается, алгоритм начинает выполнять в локальный поиск во все уменьшающейся окрестности экстремума. В задачах динамической оптимизации, напротив, лучше использовать алгоритмы роя пчел и колонии муравьев, поскольку в них предусмотрены механизмы защиты от сходимости всех частиц в окрестности одного решения: запрет на перемещение всех частиц в окрестности одного решения

и случайный поиск (алгоритм роя пчел), снижение привлекательности найденных ранее путей за счет динамического уменьшения коэффициента усиления обратной связи (алгоритм колонии муравьев). Благодаря этому алгоритмы способны быстрее находить новые решения при изменении условий задачи.

## 2.3. Адаптация алгоритмов под условия задач

### 2.3.1. Интерфейс между алгоритмами и оптимизационными задачами

Схема взаимодействия алгоритмов РИ и модели оптимизируемого объекта описана в подразделе 2.1.4 и показана на рисунке 2.1. Однако для эффективной адаптации роевых алгоритмов под специфику задачи и их простой интеграции с моделями оптимизируемых объектов необходимо определить принципы преобразования положений частиц в конкретные решения оптимизационной задачи или альтернативные варианты управления объектом.

В описаниях роевых алгоритмов использовано пространство поиска решений от 0 до 1 по всем управляемым параметрам задачи. Исключение составляет алгоритм колонии муравьев, поскольку он ищет решения на графе, а не в непрерывном пространстве поиска решений. Эвристические параметры (коэффициенты) роевых алгоритмов оказывают различное влияние на их работу в зависимости от диапазона допустимых значений позиций частиц в пространстве поиска решений. Например, для алгоритма роя светлячков существует нелинейная зависимость силы притяжения частиц от расстояния между ними. Таким образом, если диапазон значений управляющих переменных от 0 до 100, то расстояние от середины (50) до верхней границы (100) будет равно 50. Если же диапазон измерять в других единицах, он может быть, например, от 0 до 1, и то же расстояние окажется равным 0,5, в результате чего поведение частиц окажется совершенно другим для точно такой же, по сути, оптимизационной задачи. Можно привести подобный пример из алгоритма роя пчел. Коэффициенты *rad* и *rx* в алгоритме роя пчел задают, соответственно, расстояния рассеивания частиц и допустимое расстояние между областями, в которые отправляются частицы. Очевидно, что при увеличении

размера области допустимых решений эффективность алгоритма изменится, если не увеличить на ту же меру и указанные коэффициенты. В результате, параметры алгоритма придется перенастраивать даже в случае, если всего лишь изменились единицы измерения величин в модели (часы вместо секунд, километры вместо миль и т.п.).

Более того, если пространство поиска решений имеет различные ограничения по разным направлениям, то необходимо будет использовать особые значения коэффициентов по каждому направлению. Это существенно повысит трудоемкость задачи подбора значений коэффициентов. Чтобы избежать этого, в данной работе предложено использовать пространство поиска решений от 0 до 1 по всем направлениям и переводить координаты положений частиц в значения управляемых оптимизируемых переменных. В простейшем случае это можно быть выполнено следующим образом. Пусть позиция частицы

$$X = \{x_1, x_2, \ldots, x_l\},$$

а вектор управляемых переменных

$$Y = \{y_1, y_2, \ldots, y_l\}.$$

В случае системы ограничений

$$a_i \le y_i \le b_i, I = 1, \ldots, l\ , \tag{2.4}$$

получим отображение

$$y_i = x_i\,(b_i - a_i) + a_i, I = 1, \ldots, l. \tag{2.5}$$

Или, используя векторную запись:

$$Y = X \circ (B - A) + A. \tag{2.6}$$

В проектировании и управлении техническими системами часто встречаются и более сложные случаи. Они рассмотрены в разделах 3 и 4 на примерах оптимизационных задач из области электроэнергетики и календарного планирования. В задачах оптимизации электроэнергетических систем часто область допустимых решений неизвестна, поскольку определить, удовлетворяет ли

ограничениям некоторое решение, можно только после выполнения расчетов системы, например, потокораспределения в сети. Таким образом, одно и то же значение одной управляемой переменной может как удовлетворять, так и не удовлетворять ограничениям, в зависимости от значений других переменных.

В задачах планирования необходимо преобразовать позицию частицы в непрерывном пространстве поиска в дискретный календарный план. Предложенный способ использования единообразного для алгоритмов РИ пространства поиска позволяет минимизировать зависимости между алгоритмом оптимизации и оптимизируемым объектом, что повышает гибкость и простоту применения роевых алгоритмов, и является необходимым условием для эффективной адаптации путем мета-оптимизации.

### 2.3.2. Мета-оптимизация

Вторым аспектом адаптации является настройка эвристических параметров алгоритмов РИ под особенности определенного класса задач оптимизации. Согласно NFL-теореме [Wolpert, 1997; Wolpert, 2013], нельзя создать алгоритм, одинаково хорошо работающий на всем множестве возможных задач. Даже различные экземпляры задач одного класса могут иметь разную структуру, поэтому найти параметры эвристических алгоритмов, которые были бы наилучшими по эффективности для всех задач или заранее предсказать значения таких параметров для конкретного экземпляра задачи невозможно. В обзоре эвристических методов А.П. Карпенко делает обобщающий вывод, что важной особенностью РИ является наличие в них свободных параметров, сильно влияющих на эффективность, однако не существует формальных рекомендаций по выбору их значений [Карпенко, 2012].

Очень часто исследователи проводят экспериментальные подборы различных наборов коэффициентов, затем останавливаются на коэффициентах, давших в среднем лучшие результаты. М. Дориго, автор муравьиного алгоритма, пишет об использовании комбинаций из нескольких фиксированных значений:

«The values tested were: $\alpha \in \{0; 0,5; 1; 2; 5\}$, $\beta \in \{0; 1; 2; 5\}$, $\rho \in \{0,3; 0,5; 0,7; 0,9; 0,999\}$ and $Q \in \{1; 100; 10000\}$» [Dorigo, 1996]. В работе [Чураков, 2006] указано, что алгоритм колонии муравьев в значительной мере зависит от параметров, которые выбираются экспериментально. В книге [Скиена, 2013] автор пишет, что настройка эвристических алгоритмов может потребовать значительно больше усилий и времени, чем реализация. Аналогичные описания встречаются во многих работах по стохастическим эвристическим алгоритмам, не только роевым [Лопатин, 2005; Плотников, 2011; Pedersen, 2010a]. Так, М. Педерсен, проводивший глубокие исследования по существующим методам улучшения алгоритма роя частиц пишет что традиционным и простейшим подходом является изменение поведенческих параметров вручную, но такой подход ограничен в своем масштабе тем фактом, что настройка проводится вручную (интуитивно), в лучшем случае по определенной методике, но слишком примитивной [Pedersen, 2010a].

Другим подходом является автоматическая настройка параметров алгоритмов с помощью различных эвристик. Их можно разделить на две группы. К первой относятся различные правила, выведенные специально для определенного алгоритма и проверенные на некотором классе задач. Их недостатком является высокая трудоемкость получения правил в случае изменения класса решаемых задач и невозможность переноса между алгоритмами. Например, существует множество исследований по управлению параметрами роя частиц в процессе работы [Pedersen, 2010a; Gandomi, 2016; Fan, 2002; Zhou, 2016]. Но приведенные в работах правила невозможно применить для других алгоритмов РИ.

Ко второй группе относятся методы мета-оптимизации, суть которых заключается в том, что задача поиска наиболее эффективных параметров алгоритма РИ рассматривается как задача оптимизации и решается с использованием отдельного метода оптимизации, который называется мета-оптимизатором. Таким образом, один алгоритм оптимизации является надстройкой, улучшающей результаты другого, настраиваемого, алгоритма в решении определенной задачи. Такой подход намного проще в реализации и применении, чем ручная или статистическая настройка параметров или динамическая настройка в процессе

решения. Кроме того, найденные значения параметров могут быть использованы непосредственно другими разработчиками в своих реализациях роевых алгоритмов.

Сложностями мета-оптимизации являются высокие требования к вычислительной мощности, необходимость настроить, в свою очередь, параметры мета-оптимизатора, неустойчивость результатов. Под неустойчивостью здесь понимается перенастройка параметров, что приводит к хорошей работе на задачах, для которых параметры были настроены, но плохой на других задачах.

### 2.3.3. Схема предложенного алгоритма адаптации

В данной работе в качестве мета-оптимизатора выбран генетический алгоритм. Как уже отмечено, согласно NFL-теореме невозможно настроить алгоритм на всевозможные задачи оптимизации. Узкая настройка алгоритма РИ и не требуется, поскольку применение мета-оптимизации для каждой решаемой задачи будет занимать слишком большое время из-за необходимости многократно вычислять критерий, то есть выполнять расчеты на модели оптимизируемой системы. В рассматриваемых задачах оптимизации на это уходит слишком много времени не только в случае оперативного управления, но даже и для проектной постановки задач. Поэтому важно найти не наилучшие параметры алгоритм РИ, а выйти из области низкоэффективных параметров. Для такой задачи ГА подходит по аналогии с естественным отбором, когда решается задача выживания, а не достижения идеала. В работе [Панченко, 2007] приводятся достоинства и недостатки ГА. Применительно к рассматриваемой задаче оптимизации указанные достоинства особенно важны:

- ГА не требует информации о поведении оптимизируемого объекта – в задаче настройки паромеров РИ такую информацию получить затруднительно;
- ГА способны выходить из локальных экстремумов;
- ГА просты в реализации – важно, что даже простые версии ГА (одноточечный кроссинговер, выбор двух родителей) показывают высокую

эффективность, ведь иначе возникнет еще более сложная задача настройки эвристик самого мета-оптимизатора.

Что касается недостатков ГА, часть из них в данной задаче не имеет большого значения:

- ГА не позволяет гарантированно найти глобальный оптимум – такая задача и не ставится перед мета-оптимизатором;

- ГА сложно применить для нахождения всех решений задачи – аналогично;

- ГА неэффективные в решении изолированных функций – задача настройки параметров алгоритма РИ не относится к таковым.

Кроме того, задача настройки параметров алгоритма РИ имеет немного управляемых переменных, порядка 3–6 параметров, которые легко закодировать вектором чисел.

Что касается параметров самого ГА, существуют рекомендации по их выбору в общем случае [Панченко, 2007]. В данной работе используется ГА с одноточечным кроссинговером, двумя родителями, вероятностью кроссинговера 80 %, вероятностью мутации 10 %, размером популяции 50 хромосом.

В итоге используемый способ мета-оптимизации можно описать следующим образом: задача оптимизации решается алгоритмом роевого интеллекта, а генетический алгоритм используется как мета-оптимизатор, выполняющий настройку параметров алгоритма роевого интеллекта, которые используются как гены. Один набор значений параметров одной хромосоме. Полученное среднее значений критерия оптимизации используется в качестве степени приспособленности данной хромосомы. Таким образом происходит имитация естественного отбора значений параметров алгоритма РИ.

Алгоритм адаптации можно записать в виде следующего псевдокода.

1. Создать популяцию хромосом со случайными значениями параметров алгоритма РИ:

$$chromosome_i = random(p_{min\,j}, p_{max\,j});\ i = 1,\ldots, n_{ga};\ j = 1, \ldots, n_p$$

2. Выполнить $m_{ga}$ раз:

2.1. Рассчитать приспособленность, выполнив решение обучающих задач алгоритмом РИ:

$$fitness_i = \frac{1}{k}\sum_{j=1}^{k} run\_swarm(chromosomei), i = 1,..,n_{ga}$$

2.2. Сохранить наилучший набор параметров, если он был найден:

**if** *fitness*$_i$ < *best_fitness*

*best_fitness* = *fitness*$_i$

*best_parameter* = *chromosome*$_i$

2.3. Выполнить стандартные операторы генетического алгоритма:

селекцию, скрещивание, мутацию.

3. Сохранить набор параметров *best_parameter* как результат решения задачи.

В приведенном псевдокоде в пункте 2.1 подчеркнуто, что он применятся не для одной конкретной задачи, а для нескольких задач определенного класса (их можно назвать обучающими). Это позволяет избежать перенастройки на конкретную задачу, в результате которой полученные значения параметров алгоритма РИ окажутся малоэффективными для отличающихся задач.

Схема работы мета-оптимизации показана на рисунке 2.2.

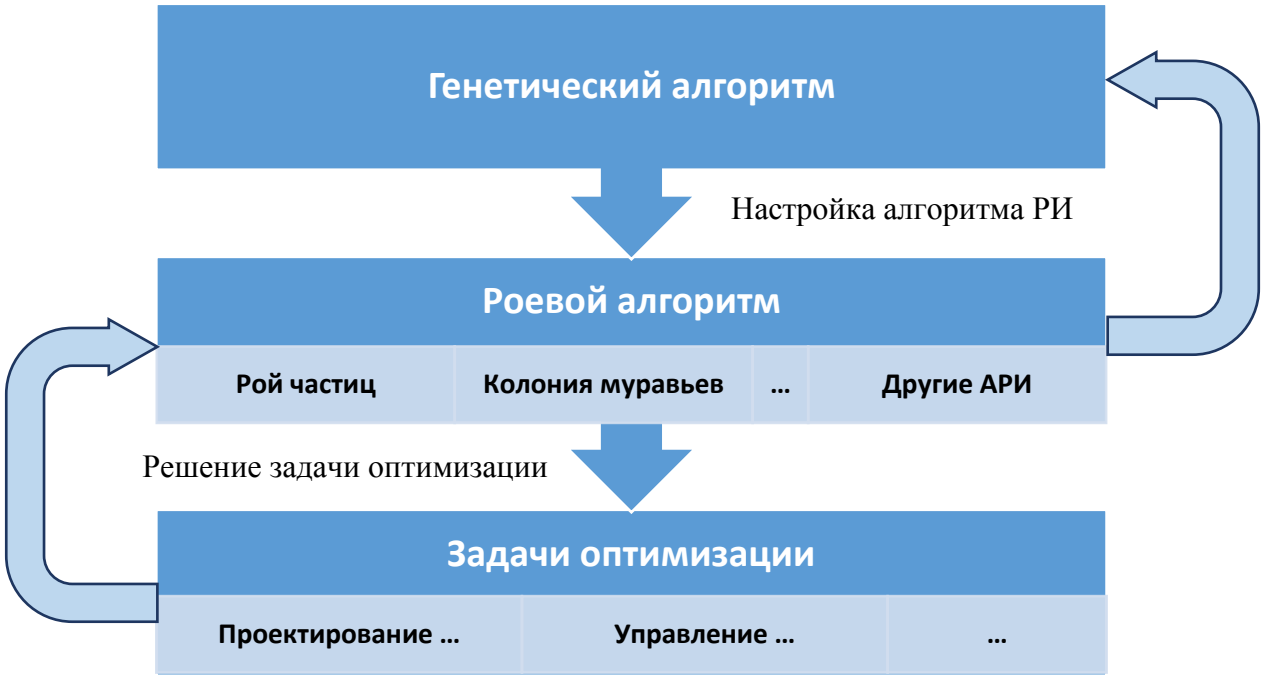



Рисунок 2.2 – Схема эволюционной мета-оптимизации

В совокупности предложенный подход к адаптации (интерфейс плюс мета-оптимизация) обладает следующими свойствами:

- может быть использован для различных классов оптимизационных задач без изменений алгоритмов оптимизации;
- использует принципы и роевого интеллекта, и эволюционной адаптации;
- может быть применен без изменений для всех алгоритмов роевого интеллекта.

Главным недостатком является необходимость длительных вычислений, хотя с развитием облачных вычислений этот недостаток можно обойти путем распараллеливания расчетов на уровне генетического алгоритма.

## 2.4. Повышение эффективности применения роевых алгоритмов

В данном подразделе описываются приемы и рекомендации, обеспечивающие повышение быстродействия алгоритмов РИ, повышение качества получаемых решений, а также вопрос организации взаимодействия программных комплексов, использующих алгоритмы РИ для решения задач оптимизации и их пользователей.

### 2.4.1. Выбор используемых структур данных

Общеизвестно, что в программировании очень важен правильный выбор структур данных. Можно реализовать набор частиц с помощью структур или матриц для повышения скорости работы программы и экономии памяти. Заметим, что это относится и к самой структуре данных для хранения частиц, когда необязательно применять массив или вектор. Можно использовать связный список, а для алгоритма роя пчел имеет смысл применить такую структуру данных, как пирамида, потому что в нем на каждой итерации требуется определять набор частиц с наилучшим значением фитнесс-функции.

В алгоритме колонии муравьев сводится к многократному обходу графа и изменению феромона на его дугах, поскольку выбор структуры графа очень важен.

На первый взгляд, может показаться, что естественной реализацией графа будет набор узлов, содержащих список дуг, каждая из которых содержит указатель на соседний узел и количественные характеристики (вес и количество феромона). Однако легко заметить, что граф можно представить в виде матрицы весов и матрицы феромона. Такой вариант будет, во-первых, проще в реализации, во-вторых, меньше по объему требуемой памяти (не нужно хранить списки указателей в каждом узле), в-третьих, скорее всего, алгоритм с таким графом будет работать быстрее.

Например, для задачи коммивояжера размерностью $N$ пунктов матрицы будут иметь размер $N \times N$, при этом значение элемента будет равно весу дуги между пунктами $i$ и $j$ для матрицы весов и количеству феромона для матрицы феромона. Имеется два варианта реализации такой матрицы: двумерный массив или одномерный. В первом случае для доступа к элементам необходимо использовать два индекса $i, j$. Во втором – один, для элемента рассмотренной выше матрицы, в этом случае индекс можно записать как $N \times i + j$ и этом вариант лучше по быстродействию. Использование адресной арифметики, то есть операций с указателями вместо индексации (если выбранная система программирования позволяет) дополнительно повысит скорость работы. В этом случае многие операции, требующие обхода всех дуг графа в произвольном порядке (испарение феромона, очистка, инициализация) могут быть проведены простым перемещением указателя по всему массиву с выполнением нужных действий над каждым элементом.

### 2.4.2. Учет постоянной составляющей критерия оптимальности

Во многих алгоритмах оптимизации значение критерия используется не только для качественного, но и для количественного управления алгоритмом. Например, в формулах для алгоритма роя частиц, светлячков и пчел $f(X)$ используется только качественно для определения лучшей позиции в пространстве поиска решений. А в формулах для алгоритма колонии муравьев от значения

критерия пропорционально зависит количество феромона, наносимое на граф, следовательно, и вероятности выбора ребер графа.

Необходимо учитывать, что критерий оптимальности может иметь некоторую постоянную составляющую $f_{const}$, значение которой намного больше, чем диапазон изменяемой части $f_{var}(X)$, т. е.

$$f(X) = f_{const} + f_{var}(X)$$

В этом случае относительные различия между значениями фитнесс-функций частиц малы. Если постоянная составляющая в 100 раз больше диапазона изменяемой части, то максимально возможная разница между фитнессами частиц составит всего 1 %.

Из двух приведенных выше абзацев следует простой вывод: если критерий количественно влияет на процесс работы алгоритма и его постоянная часть намного превышает изменяемую часть, то эффективность алгоритма может очень сильно упасть. Например, для генетического алгоритма такая ситуация приведет к тому, что вероятность отбора всех частиц на этапе селекции будет примерно одинаковой независимо от качества их позиций в пространстве поиска решений. А в алгоритме колонии муравьев количество феромона, наносимое на ребра графа, будет также почти одинаковым независимо от эффективности найденного каждой частицей маршрута. Таким образом, алгоритмы лишаются одного из своих главных свойств – учета опыта, полученного на предыдущих итерациях. По этой причине применение эволюционной мета-оптимизации на практике может быть малоэффективным, без применения описанного здесь учета постоянной составляющей.

Для устранения этого негативного эффекта можно использовать различные способы. В общем случае можно на первой итерации выбрать наименьшее найденное решение, принять его за примерную постоянную величину ($f_{appr}$) и затем вычитать его на всех итерациях из полученных значений критерия. Либо после окончания работы алгоритма сохранить наилучшее значение критерия, и при следующем запуске вычитать это значение для определения фитнессов частиц. Нужно учесть, что не все алгоритмы корректно работают с отрицательными

значениями критерия, которые могут появляться в данном подходе, если фитнесс какой-либо частицы окажется меньше величины $f_{appr}$.

### 2.4.3. Взаимодействие с пользователем

При разработке приложений, в которых решаются задачи оптимизации, необходимо учитывать процедуру взаимодействия пользователя и приложения во время выполнения длительных расчетов. Во-первых, необходимо дать пользователю возможность остановить расчеты и корректно завершить работу приложения. Во-вторых, желательно сообщать информацию о ходе решения. Можно выделить несколько вариантов:

- приложение никак не взаимодействует с пользователем во время решения задачи;

- приложение с определенной частотой переключается с процесса решения задачи на процесс вывода информации для пользователя и обработки его действий;

- вычисления и организация диалога с пользователем разделены на два параллельно работающих потока.

Первый вариант является самым эффективным с точки зрения скорости расчетов, так как программа выполняет вычисления без затрат времени на взаимодействие с пользователем. Этот вариант наиболее просто реализовать. Второй вариант намного уступает по времени решения задач из-за многочисленных пауз в расчетах, но во время этих остановок можно выполнять необходимые для взаимодействия с пользователем операции.

Третий вариант является предпочтительным, поскольку почти не уступает первому по скорости (в случае многоядерной архитектуры) и предоставляет необходимые для пользователя функции вывода сообщений о ходе расчетов и возможности преждевременного, но корректного завершения.

### 2.4.4. Использование параллельных вычислений

Поскольку алгоритмы РИ основаны на использовании множеств простых частиц, можно предполагать, что их работу можно выполнять параллельно на нескольких процессорах или ядрах одного процессора. Распараллеливание работы алгоритмов РИ выполняется просто, поскольку перемещение каждой частицы в пределах одной итерации происходит независимо, как и вычисление критерия по каждой из них. Иными словами, простота параллельной работы алгоритмов РИ объясняется использованием принципа косвенного обмена информацией между частицами. Чем проще косвенный обмен, тем проще и распараллеливание алгоритма РИ. В алгоритме роя частиц распараллеливание происходит наиболее просто и эффективно, поскольку средством косвенного обмена $M$ является вектор координат наилучшего найденного решения, который используется всеми частицами для чтения, а для записи только в редких случаях нахождения лучшего решения.

Распараллеливание алгоритма колонии муравьев значительно более сложная задача. Обход графа каждой частицей и вычисление критерия происходит независимо от других частиц в рамках одной итерации. Инициализация графа и испарение феромона происходит для каждой дуги независимо от остальных дуг. Поэтому перечисленные этапы легко и эффективно поддаются распараллеливанию, но этап нанесения феромона требует более аккуратного распараллеливания, как показывает следующий пример. Допустим, на некотором ребре графа имеется $\tau$ феромона, и две частицы, управляемые двумя потоками, одновременно пытаются нанести на это ребро количество феромона $\tau_1$ и $\tau_2$, соответственно. В результате новое значение феромона должно стать $\tau + \tau_1 + \tau_2$, но при этом может возникнуть следующая ситуация:

- частица 1 прочитала значение $\tau$;
- частица 2 прочитала значение $\tau$;
- частица 1 вычислила $\tau_1' = \tau + \tau_1$ и записала это значение вместо $\tau$;
- частица 2 вычислила $\tau_2' = \tau + \tau_2$ и записала это значение вместо $\tau_1'$.

В результате новое значение феромона на ребре станет $\tau + \tau_2$, а данные о том, что первая частица прошла по данному ребру, просто пропадут. Это может сильно снизить эффективность алгоритма, ведь могут пропасть данные о близком к оптимальному пути. Для решения этой проблемы можно либо отказаться от параллельного выполнения этого этапа, либо использовать механизм блокировок, то есть пока один поток выполняет чтение и перезапись феромона на ребре, доступ к ребру блокируется для всех остальных потоков. Выбор варианта зависит от соотношения размера графа и числа частиц. Например, при большом размере графа и малом числе частиц вероятность подобных конфликтов доступа низка, и второй вариант будет предпочтительнее. При большом же количестве частиц слишком большое количество процессорного времени будет тратиться на управление доступом. По закону Амдала, эффективность параллельных вычислений тем выше, чем больше доля вычисления, которые могут быть распараллелены [Эхтер, 2010].

$$S(p) \leq \left(a + \frac{1-a}{p}\right)^{-1} \tag{2.7}$$

где

- $S(p)$ – теоретическое ускорение, т.е. во сколько раз увеличится скорость работы за счет параллелизма;

- $p$ – количество процессоров;

- $a$ – доля последовательных операций в общем объеме вычислений.

Из закона Амдала следует, что целесообразно вводить распараллеливание на более высоком уровне работы приложения. Поскольку данная работа использует мета-оптимизацию алгоритмов РИ, наиболее эффективно проводить распараллеливание на этом уровне. Исходная популяция разделяется на части по числу ядер, и вычисление критерия алгоритмами роевого интеллекта происходит параллельно. Отбор и скрещивание выполняются централизованно на этапе работы эволюционного мета-оптимизатора, поэтому один и тот же механизм работает независимо от используемого алгоритма РИ. При этом во время работы каждого из роев не используются общие данные, поэтому процессы легко синхронизируются. На рисунке 2.3 показано, как при повышении количества процессоров или ядер

повышается средняя скорость работы алгоритмов РИ. При этом эффективность тем выше, чем больше размерность задачи и чем ниже быстродействие алгоритма РИ.

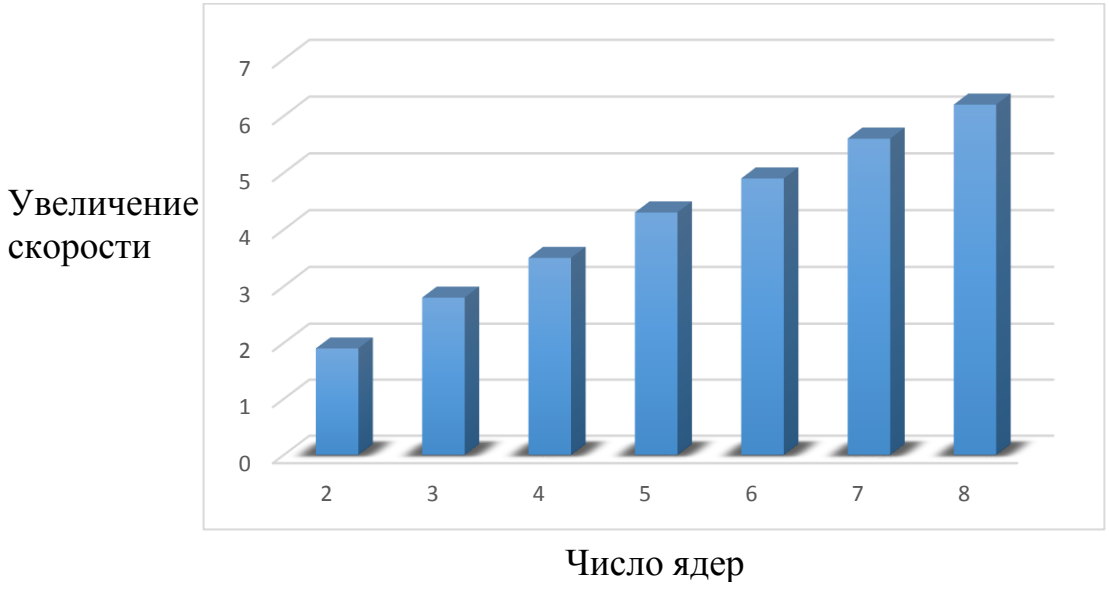



Рисунок 2.3 – Эффективность распараллеливания расчетов адаптивных алгоритмов РИ

## 2.5. Выводы по разделу 2

Проведен системный анализ алгоритмов роевого интеллекта и разработана новая модель описания, реализации и применения роевых алгоритмов. Введена и обоснована единая для всех роевых алгоритмов терминология. Описаны общие для них принципы и схема работы. Это позволило выделить общие и уникальные для каждого алгоритма особенности и приемы, обеспечивающие их большую или меньшую эффективность в различных классах оптимизационных задач. Показано, что каждый алгоритм роевого интеллекта может быть описан путем задания набора частиц $S$, средства для косвенного обмена опытом $M$, правил перемещения частиц $A$, эвристических параметров $P$, а также интерфейса для взаимодействия с решаемой задачей $I$, $O$ (входной и выходной каналы). Предложенная модель алгоритмов роевого интеллекта обеспечивает стандартизацию, повышение гибкости и переносимости создаваемого программного обеспечения, повышение скорости разработки, а также создание новых гибридных алгоритмов роевого

интеллекта путем выбора нужной комбинации особенностей различных алгоритмов.

Алгоритмы роевого интеллекта следует выделить в обособленную группу среди популяционных алгоритмов, поскольку они имеют общие базовые принципы работы, основанные на коллективных децентрализованных перемещениях частиц и косвенном обмене информации, в отличие от принципов отбора, по которым работают эволюционные алгоритмы.

Обосновано, что эффективная адаптация алгоритмов роевого интеллекта имеет два основных аспекта: отсутствие зависимостей между реализаций алгоритмов и моделью оптимизируемой системы (решаемой задачей); настройка значений эвристических коэффициентов роевых алгоритмов под определенный класс задач. Для устранения зависимостей между алгоритмом и решаемой задачей между ними введен интерфейс, позволяющий выполнить их соединение без необходимости менять что-либо в математических моделях алгоритмов или в моделях оптимизируемых систем. Необходимые особенности, такие как масштабирование значений управляемых переменных, учет ограничений и дискретность, учитываются в интерфейсе. Задача оптимизации становится для алгоритма черным ящиком, как и алгоритм для задачи. Благодаря этому единая реализация алгоритма может применяться для задач непрерывной оптимизации, задач дискретной оптимизации, гибридных задач, для задач условной или безусловной оптимизации.

Для настройки эвристических параметров роевых алгоритмов под решаемые задачи использована мета-оптимизация на базе генетического алгоритма по аналогии с эволюционной адаптацией в природе. В данной работе новизна заключается в повышении эффективности мета-оптимизации за счет отделения алгоритма роевого интеллекта от оптимизационной задачи. В противном случае перенастройку параметров роевого алгоритма путем мета-оптимизации приходилось применять даже при незначительных изменениях условий задачи.

Изложены подходы и рекомендации, позволяющие повысить эффективность и быстродействие роевых алгоритмов при их программной реализации.

# 3. Адаптивные алгоритмы роевого интеллекта в задачах оптимизации в электроэнергетике

В третьем разделе приведены результаты решения оптимизационных задач в области электроэнергетики. Потребление энергоресурсов с каждым годом растет, по прогнозам, приведенным в [Семенов, 2014], к 2050 году оно увеличится вдвое. Как показано в [Artunov, 2017], даже полный переход на возобновляемые источники энергии не решит экологических проблем: использование энергии ветра требует больших площадей для ветроэлектростанций, нарушает климатические условия, создает акустический шум; гидроэлектростанции требуют затопления больших территорий, нарушают миграцию рыб; солнечная энергетика также требует выделения площадей и может негативно влиять на климат. Поэтому эффект от снижения потерь электроэнергии нужно рассматривать не только с точки зрения финансовой выгоды, но и с точки зрения экологии.

В работе решаются различные классы задач (непрерывные, комбинаторные, гибридные; статические и динамические), чтобы продемонстрировать эффективность созданного инструмента оптимизации на основании адаптивных алгоритмов роевого интеллекта. Раздел состоит из трех подразделов и заключения. Подраздел 3.1 описывает применение адаптивных алгоритмов роевого интеллекта для решения задачи оптимизации источников реактивной мощности (их расположения и мощностей) для глубокой компенсации реактивной мощности. Задача представляет как практический, так и исследовательский интерес, имеет несколько критериев, высокую вычислительную сложность, наличие и непрерывных, и дискретных свойств, а также может быть рассмотрена и как задача проектирования, и как задача оперативного управления. Подраздел 3.2 посвящен оптимизации коэффициентов трансформации, характерной для электроэнергетики комбинаторной задаче, которая возникает и на этапе проектирования, и во время управления электроэнергетическими системами. В подразделе 3.3 приводится пример использования результатов диссертационной работы для оптимизации моделей предсказания графиков нагрузки.

## 3.1. Оптимизация размещения источников реактивной мощности

Вопросы компенсации реактивной мощности подробно освещены в работах [Глушков, 1975; Кабышев, 2012; Манусов, 2017b]. Одним из наиболее важных источников потерь электроэнергии является передача реактивной мощности в линиях систем электроснабжения, особенно в условиях большой протяженности линий электропередач. Задача актуальна в сетях различного класса напряжений: от систем электроснабжения отдельных предприятий до энергосистемы страны. Передача реактивной мощности снижает эффективность электроснабжения, поскольку за ее счет не выполняется полезная работа, но протекание реактивного тока в линиях электропередач занимает часть сечения кабеля, снижая пропускную способность. Это, в свою очередь, приводит к потерям активной энергии, поскольку потери мощности пропорциональны квадрату полного тока. Влияние реактивной мощности на потери активной мощности можно определить следующим образом:

$$\Delta P = 3I^2R = \frac{S^2}{U^2}R = \frac{P^2+Q^2}{U^2}R = \frac{P^2}{U^2}R + \frac{Q^2}{U^2}R$$

Компенсация реактивной мощности влияет не только на экономию расходов на электроэнергию, но и на срок службы оборудования за счет снижения электрической нагрузки и теплового выделения, в частности в трансформаторах.

В работе [Красник, 2004] приводятся следующие негативные эффекты от повышенного потребления реактивной мощности:

1. Из-за увеличения передаваемого тока

$$I = \frac{\sqrt{P^2+Q^2}}{\sqrt{3}\,U}$$

ухудшается пропускная способность линий, что приводит к увеличению сечения проводников и необходимости прокладки новых магистралей.

2. Из-за увеличения потерь активной мощности

$$\Delta P = \frac{P^2+Q^2}{U^2}R$$

перерасходуется электроэнергия.

3. Из-за увеличения потерь напряжения

$$\Delta U = \frac{PR + QX}{U}$$

снижается частота вращения асинхронных двигателей, уровень освещенности, что приводит к снижению производительности людей и оборудования, ухудшению качества выпускаемой продукции.

### 3.1.1. Постановка задачи оптимизации источников реактивной мощности

Одним из способов повышения энергоэффективности является компенсация реактивной мощности. При этом эффективнее использовать глубокую компенсацию, когда компенсирующие установки (КУ, источники реактивной мощности) располагаются вблизи ее основных потребителей.

Общей целью является выработка решения, которое обеспечит максимальный экономический эффект при соблюдении условий нормальной работы как электрических сетей, так и приемников электроэнергии [Железко, 1981]. Возникает многокритериальная дискретно-непрерывная задача выбора узлов для размещения компенсирующих установок и выбора их мощностей [Манусов, 2017b]. Задача оптимизации имеет два критерия: снижение потерь активной мощности и минимум затрат на компенсирующие установки. Наиболее простым способом сведения многокритериальной задачи к однокритериальной считается линейная свертка с применением весовых коэффициентов. В данном случае и потери активной мощности, и затраты на компенсирующие установки можно объединить в один критерий – финансовые затраты *W*. Критерий будет учитывать стоимость единицы потерь активной мощности и стоимость единицы мощности компенсирующей установки. Стоимость потерь активной мощности определяется тарифом, а стоимость единицы мощности компенсирующей установки определяется как среднее отношение цены компенсирующей установки к ее мощности [Третьякова, 2015]. Кроме того, поскольку энергосистема должна сохранять устойчивость, к критерию добавлена штрафная функция. Если для

варианта размещения компенсирующих установок условие устойчивости не выполняется, то к значению критерия оптимизации прибавляется штраф, значение которого на порядок больше, чем возможное значение критерия.

Математическая формулировка однокритериальной задачи оптимизации сформулирована следующим образом:

$$W(Q) = c_{cu} \sum_{i=1}^{n} Q_i + c_p t \Delta P(Q) + G(Q) \rightarrow min \qquad (3.1)$$

при ограничениях: $Q_{min\,i} < Q_i < Q_{max\,i}$ , где

$Q_i$ – мощность КУ в $i$-м узле, при этом значение $Q_{max\,i} = 0$ показывает, что в $i$-м узле невозможно или же не рассматривается размещение КУ;

$c_{cu}$ – средняя цена за единицу мощности компенсирующей установки;

$c_p$ – цена за кВт·ч активной мощности;

$n$ – количество узлов, в которые можно поставить КУ;

$t$ – расчетный период в часах;

$\Delta P(Q)$– суммарные потери активной мощности в сети;

$G(Q)$ – штрафная функция;

$i$ – индекс компенсирующей установки: $i = 1, 2, 3, \ldots, n$.

### 3.1.2. Применение адаптивных роевых алгоритмов для оптимизации источников реактивной мощности

Сложностью данной задачи является высокая вычислительная сложность расчета критерия оптимальности и высокая трудоемкость аналитического решения данной оптимизационной задачи. Поэтому наиболее эффективным является предлагаемый в данной работе подход по рассмотрению оптимизируемого объекта как черного ящика, который исследует адаптивный алгоритм роевого интеллекта, не проникая внутрь, как показано на рисунке 2.1.

Для применения РИ необходимо установить соответствие между позицией частицы $X$ и вектором-решением задачи $Q$. В случае сложной топологии сети максимальная возможная для компенсации реактивная мощность в каждом

отдельном узле зависит от компенсации в других узлах и заранее неизвестно, так что невозможно заранее определить границы пространства поиска решений. Поэтому вектор $X$ использовался как вектор коэффициентов, так что мощность каждой компенсирующей установки определялась следующим образом:

$$Q_{ij} = \begin{cases} x_{ij}Q_{max\,ij}, x_{ij}Q_{max\,ij} \geq Q_{min} \\ 0, x_{ij}Q_{max\,ij} < Q_{min} \end{cases}, i = 1, \dots, n \quad (3.2)$$

Где помимо введенных ранее обозначений $j$ – номер итерации алгоритма, $Q_{min}$ – величина минимальной возможной мощности компенсирующей установки, необходимая во избежание затрат на установку близких к нулю реактивных мощностей.

### 3.1.3. Вычислительные эксперименты в задаче оптимизации источников реактивной мощности

Эксперименты проводились на математической модели системы электроснабжения подстанции предприятия «Ангарский электролизный химический комбинат». В каждой секции подстанции имеется 14 узлов для потенциального размещения КУ в узлах с распределительными шкафами. Основными потребителями энергии являются электронасосы, система вентиляции, система подкачивающих компрессоров. Результаты экспериментов для алгоритма роя частиц (обозначен РЧ без адаптации и АРЧ с адаптацией) и алгоритма роя пчел (РП и АРП, соответственно) приведены в таблицах 3.1 и 3.2. В таблице 3.2 и на рисунке 3.1 показано, как повышается эффект от внедрения глубокой компенсации с течением времени.

Эксперимент показал, что использование адаптивных алгоритмов РИ позволяет снизить потери активной мощности на 19,6 % и финансовые затраты на 8,6 % для периода работы 4 года, что составляет 289 тысяч рублей для одной секции, что соответствует 2,312 млн. рублей для всей системы. Разность между снижениями потерь при использовании РИ без адаптации и с адаптацией равна 8,65 % за расчетный период 4 года.

Алгоритмы роя частиц и роя пчел показали близкие результаты, наилучшее решение найдено алгоритмом роя частиц. Тем не менее априори эффективность роевых алгоритмов невозможно сопоставить, поэтому для достижения как можно более высокого качества решения следует применять различные алгоритмы. Как будет показано ниже, в данной задаче при модификации условий (динамической оптимизации) алгоритм роя частиц проиграет сравнение алгоритму роя пчел.

Таблица 3.1 – Результаты алгоритмов в задаче оптимального распределения КУ

| **Алгоритм** | **Критерий задачи (3.1) *W*, тыс. руб.** | **Потери активной мощности Δ*P*, кВт** | **Суммарная мощность компенсаторных установок, кВар** | **tg(φ)** |
|---|---|---|---|---|
| Без оптимизации | 3353 | 40,55 | 0 | 0,55 |
| РЧ | 3082 | 33,17 | 1129,8 | 0,138 |
| АРЧ | 3064 | 32,59 | 1231,3 | 0,1 |
| РП | 3087 | 33,22 | 1132,7 | 0,12 |
| АРП | 3068 | 32,64 | 1230,0 | 0,1 |

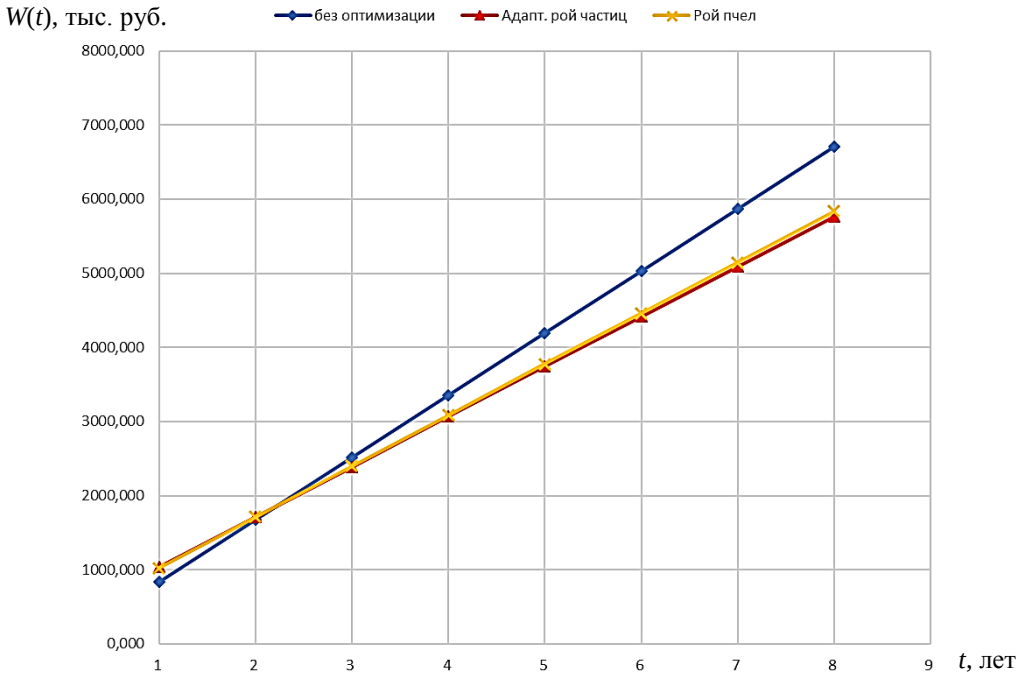



Рисунок 3.1 – Результаты алгоритмов в краткосрочной и долгосрочной перспективе в задаче оптимального распределения КУ

Таблица 3.2 – Результаты в краткосрочной и долгосрочной перспективе

| **Период** | **Финансовый эффект от компенсации для всей системы электроснабжения, тыс. руб.** | | | |
|---|---|---|---|---|
| | **РЧ** | **АРЧ** | **РП** | **АРП** |
| 1 | -1491 | -1639 | -1506 | -1644 |
| 2 | -270 | -322 | -294 | -336 |
| 3 | 950 | 994 | 918 | 973 |
| 4 | 2171 | 2311 | 2131 | 2281 |
| 5 | 3391 | 3627 | 3343 | 3589 |
| 6 | 4612 | 4944 | 4555 | 4897 |
| 7 | 5832 | 6260 | 5768 | 6206 |
| 8 | 7053 | 7577 | 6980 | 7514 |

### 3.1.5. Оптимизация размещения установок компенсации реактивной мощности в сетях 0,4 кВ электроснабжения АО «УЭХК»

Для АО «Уральский электрохимический комбинат» был выполнен расчет коэффициента мощности ($tg(\varphi)$) на щитах собственных нужд напряжением 0,4 кВ подстанции Р-12 (далее по тексту – ЩСН-0,4 кВ п/ст. Р-12) и определение наилучшего варианта размещения установок компенсации реактивной мощности. Основной целью работы являлось определение потерь активной мощности в кабельных линиях схемы электроснабжения потребителей п/ст. Р-12 от передачи реактивной мощности и нахождение оптимального варианта размещения установок компенсации реактивной мощности (УКРМ) в распределительной электрической сети с применением адаптивного алгоритма роевого интеллекта и проверка экономического эффекта от применения глубокой компенсации реактивной мощности. При этом необходимо рассмотреть УКРМ с возможностью регулирования и без оной. Результаты расчетов приведены в таблицах 3.3–3.7, при этом введены следующие обозначения: $\Delta P$ – потери активной мощности в кабельных линиях, кВт; $W$ – ежегодная стоимость потерь активной мощности в рублях; $SQ$ – суммарная мощность конденсаторных установок, кВАр.

Экспериментально на модели показано, что глубокая компенсация реактивной мощности позволит предприятию снизить потери электроэнергии при

передаче ее по электросетям 0,4 кВ на 1.7 кВт (22,2 %) как при установке нерегулируемых УКРМ, так и при установке регулируемых УКРМ.

Таблица 3.3 – Значения параметров секций без УКРМ

| **Номер секции** | **tg(φ)** | **Δ*P*** | ***W*** |
|---|---|---|---|
| 1 секция | 0,574076 | 4,37969 | 93447,1 |
| 2 секция | 0,622325 | 3,24849 | 69311,3 |

Таблица 3.4 – Значения параметров секций с УКРМ без регулирования

| **Номер секции** | **tg(φ)** | **Δ*P*** | ***W*** |
|---|---|---|---|
| 1 секция | 0,1 | 3,444 | 73488 |
| 2 секция | 0,1 | 2,49284 | 53188,4 |

Таблица 3.5 – Значение затрат на УКРМ без регулирования

| **Номер секции** | ***SQ*** | **Δ*P*** | ***W*** |
|---|---|---|---|
| 1 секция | 129,641 | 51856,4 | 19959,1 |
| 2 секция | 113,31 | 45324,1 | 16123 |

Таблица 3.6 – Значения параметров секций с регулируемыми УКРМ

| **Номер секции** | **tg(φ)** | **Δ*P*** | ***W*** |
|---|---|---|---|
| 1 секция | 0,113 | 3,4525 | 73664,2 |
| 2 секция | 0,152 | 2,5178 | 53721 |

Таблица 3.7 – Значение затрат на регулируемые УКРМ

| **Номер секции** | ***SQ*** | **Δ*P*** | ***W*** |
|---|---|---|---|
| 1 секция | 126,181 | 69399,3 | 19782,9 |
| 2 секция | 102,009 | 56154,5 | 15590,3 |

Снижение потерь мощности в годовом исчислении составит 14,9 МВт·ч или:

- для нерегулируемых УКРМ 36,1 тыс. руб. со сроком окупаемости в среднем 2.7 года;

- для регулируемых УКРМ 35,4 тыс. руб. со сроком окупаемости в среднем 3,55 года.

Проведенная научно-исследовательская практическая работа проведена для

секции системы электроснабжения АО «Уральский электрохимический комбинат», получен акт о внедрении.

### 3.1.6. Оперативное управление источниками реактивной мощности

**Постановка задачи.** Приведенная задача (3.1) может быть рассмотрена и как задача оперативного управления мощностями КУ уже после их размещения в узлах сети, то есть как задача динамической оптимизации, что особенно важно в связи с развитием автоматизации управления режимами электроэнергетических систем [Руденко, 2000; Семенов, 2014; Филиппова, 2016] в настоящее время. В диссертационной работе проведено исследование влияния динамических изменений условий задачи на эффективность роевых алгоритмов и определение наилучших способов учета таких изменений. Чтобы провести вычислительные эксперименты, использовалось моделирование выхода из строя компенсирующих установок. Задачей алгоритма оптимизации в этом случае является автоматическая регулировка мощностей оставшихся КУ в реальном времени для обеспечения устойчивости электроэнергетической системы и сокращения потерь активной мощности. Критерий оптимальности близок к критерию 3.1, но не использует суммарные мощности КУ, поскольку считается, что КУ уже расположены во определенных узлах сети:

$$W(Q) = \Delta P(Q) + G(Q) \rightarrow min \qquad (3.3)$$

**Управление популяционными алгоритмами в динамически меняющихся задачах.** Были рассмотрены два альтернативных варианта применения популяционных алгоритмов оптимизации в указанной задаче.

1. В момент изменения условий задачи завершить работу алгоритма и сразу же запустить алгоритм для новой задачи, полученной в результате изменения условий. Решение исходной задачи сводится к решению набора несвязанных задач статической оптимизации. Это позволяет популяционному алгоритму выходить из

локальных экстремумов, но накопленный ранее опыт теряется при каждом изменении условий задачи. Этот способ далее именуется «с перезапуском».

2. В момент изменения условий задачи не выполнять никаких внешних управляющих воздействий на популяционный алгоритм. В этом случае при небольших изменениях алгоритм может очень быстро найти новое оптимальное (или хотя бы удовлетворительное) решение. Однако при существенном изменении условий алгоритм может не выйти из текущего окрестности. Этот способ далее именуется «без перезапуска».

Исходя из приведенных во втором разделе свойств роевых алгоритмов, был сделан вывод, что наилучшим для таких задач является алгоритм роя пчел. Поскольку он в каждый момент работы гарантированно хранит несколько удаленных друг от друга решений. В отличие от алгоритма роя частиц, который с течением времени сходится в окрестности одного экстремума. Также для сравнения был рассмотрен и ГА. По результатам теоретического анализа до проведения экспериментов была выдвинута гипотеза, что путь без перезапуска предпочтительнее для алгоритмов роя частиц и генетического алгоритма, а алгоритм роя пчел эффективнее без перезапуска, так как из трех указанных алгоритмов только алгоритм роя пчел имеет специальные средства, запрещающие большинству частиц находиться в небольшой окрестности одного решения.

**Эксперимент** заключался в том, что каждый алгоритм, работающий с перезапуском или без него, находил решение оптимизационной задачи, после чего условия задачи менялись путем моделирования отказа определенной КУ, чтобы алгоритм решил задачу с изменившимися условиями. Схема проведения вычислительного эксперимента приведена ниже.

1. Запустить алгоритм оптимизации для решения задачи (3.3) с начальными условиями.

2. По-очереди для каждого узла с КУ.

2.1. Смоделировать отказ КУ.

2.2. Если алгоритм работает с перезапуском, то запустить его заново.

2.3. Остановить алгоритм через 200 тысяч итераций после отказа КУ.

Поскольку данные задачи часто требуется решать в режиме реального времени, очень важна скорость нахождения решения. Поэтому производилась запись полученных после ста, пятисот, тысячи и двух тысяч итераций. Выполнение 200 тысяч итераций нужно было только для оценки конечного результата. В режиме реального времени это число итераций слишком велико для рассматриваемой задачи с учетом больших затрат времени на расчет критерия для каждой частицы на каждой итерации. Используемые значения алгоритмов оптимизации приведены в таблице 3.8.

Таблица 3.8 – Параметры алгоритмов, используемых в решении задачи динамической оптимизации КУ

| **Алгоритм** | **Количество частиц** | **Эвристические параметры** |
|---|---|---|
| Роя частиц | 100 | $\alpha_1 = 2{,}03$ $\alpha_2 = 2{,}32$, $\omega = 0{,}87$, $\beta = 0{,}9$ |
| Роя пчел | 100 | $n^s = 10$, $n^b = 15$, $n^g = 10$, $c^b = 4$, $c^g = 3$, $rad = 0{,}01$, $rx = 0{,}05$ |
| Генетический | 100 | $p_{xover} = 0{,}9$, $p_{mutation} = 0{,}2$ |

Результаты экспериментов приведены в таблицах 3.9–3.11. В первом столбце указан используемый алгоритм и способ меняющихся условий задачи: (П) – с перезапуском, (БП) – без перезапуска. В таблице 3.9 во втором столбце указан номер КУ, отказ которой моделировался в данном запуске эксперимента. Следующие за ним столбцы содержат значения критерия после указанного в скобках в заголовке столбца количества выполненных итераций. Результаты экспериментов, усредненные по всем КУ, приведены в таблицах 3.10 и 3.11 и показаны на рисунках 3.2 и 3.3, соответственно.

**Анализ результатов.** Проведенные вычислительные эксперименты позволяют сделать несколько выводов.

1. Алгоритм роя пчел эффективен в задачах динамической оптимизации, причем разница между алгоритмом с перезапуском и без него оказалась несущественной (0,1%). При правильно настроенных параметрах частицы алгоритма роя пчел всегда рассредоточены по нескольким экстремумам, что позволяет как учитывать ранее найденные решения, так и находить новые.

2. Хотя алгоритм роя пчел находит новое эффективное решение быстро (за первые сто итераций), затем очень медленно эффективность повышается. Возможно, дело в пчелах-разведчиках, которые выполняют полностью независимый случайный поиск, поэтому чем больше итераций выполнено, тем больше вероятность обнаружения ими нового более эффективного решения. Этот вопрос требует дополнительных исследований.

3. Необходимо применять перезапуск алгоритма роя частиц, чтобы алгоритм смог выходить из локального экстремума, в окрестности которого собираются частицы после достаточно большого числа итераций.

Таблица 3.9 – Результаты решения динамической задачи (3.2)

| **Алгоритм** | **КУ** | **ΔP(100), кВт** | **ΔP(500), кВт** | **ΔP(1000), кВт** | **ΔP(2000), кВт** | **ΔP(20000), кВт** |
|---|---|---|---|---|---|---|
| АРЧ (П) | 1 | 312,02 | 311,92 | 311,92 | 311,92 | 311,9 |
| АРЧ (БП) | 1 | 324,48 | 324,48 | 324,48 | 324,48 | 324,48 |
| АРП (П) | 1 | 311,99 | 311,42 | 310,97 | **310,7** | **306,14** |
| АРП (БП) | 1 | **311,8** | **311,37** | **310,93** | 310,77 | 310,34 |
| ГА (П) | 1 | 346,34 | 346,34 | 344,27 | 333,88 | 316,42 |
| ГА (БП) | 1 | 343,28 | 339,19 | 339,19 | 335,69 | 318,15 |
| АРЧ (П) | 7 | **311,32** | 311,31 | 311,31 | 311,31 | 311,31 |
| АРЧ (БП) | 7 | 318,06 | 318,06 | 318,06 | 318,06 | 318,06 |
| АРП с (П) | 7 | 311,54 | **311,02** | **310,87** | **310,8** | 304,43 |
| АРП (БП) | 7 | 311,36 | 311,09 | 310,96 | 310,68 | **300,57** |
| ГА (П) | 7 | 354,69 | 344,42 | 344,42 | 338,26 | 315,68 |
| ГА (БП) | 7 | 345,26 | 345,26 | 345,26 | 342,53 | 319,4 |
| АРЧ с (П) | 9 | 352,67 | 352,67 | 352,67 | 352,67 | 352,67 |
| АРЧ (БП) | 9 | 392,46 | 392,46 | 392,46 | 392,46 | 392,46 |
| АРП (П) | 9 | 324,59 | 324,09 | 323,9 | 323,55 | **315,19** |
| АРП (БП) | 9 | **324,19** | **323,54** | **323,36** | **323,15** | 321,44 |
| ГА (П) | 9 | 392,87 | 392,87 | 392,87 | 392,87 | 368,21 |
| ГА (БП) | 9 | 408,93 | 396,98 | 387,31 | 387,31 | 366,19 |
| АРЧ (П) | 12 | 358,78 | 358,78 | 358,78 | 358,78 | 358,78 |
| АРЧ (БП) | 12 | 398,97 | 398,97 | 398,97 | 398,97 | 398,97 |
| АРП (П) | 12 | 330,12 | 329,61 | 329,36 | 329,03 | **318,74** |
| АРП (БП) | 12 | **329,83** | **329,06** | **328,83** | **328,63** | 321,72 |
| ГА (П) | 12 | 409,76 | 407,65 | 399,73 | 392,86 | 382,14 |
| ГА (БП) | 12 | 415,28 | 409,39 | 408,97 | 399,58 | 377,26 |
| АРЧ (П) | 13 | 311,57 | 311,57 | 311,56 | 311,56 | 311,56 |
| АРЧ (БП) | 13 | 322,89 | 322,89 | 322,89 | 322,89 | 322,89 |
| АРП (П) | 13 | 311,58 | 311,4 | 311,33 | 311,23 | **300,08** |
| АРП (БП) | 13 | **311,07** | **310,91** | **310,91** | **310,76** | 304,81 |
| ГА (П) | 13 | 352,56 | 342,97 | 340,22 | 332,53 | 312,68 |
| ГА (БП) | 13 | 344,4 | 344,4 | 344,4 | 327,93 | 319,65 |

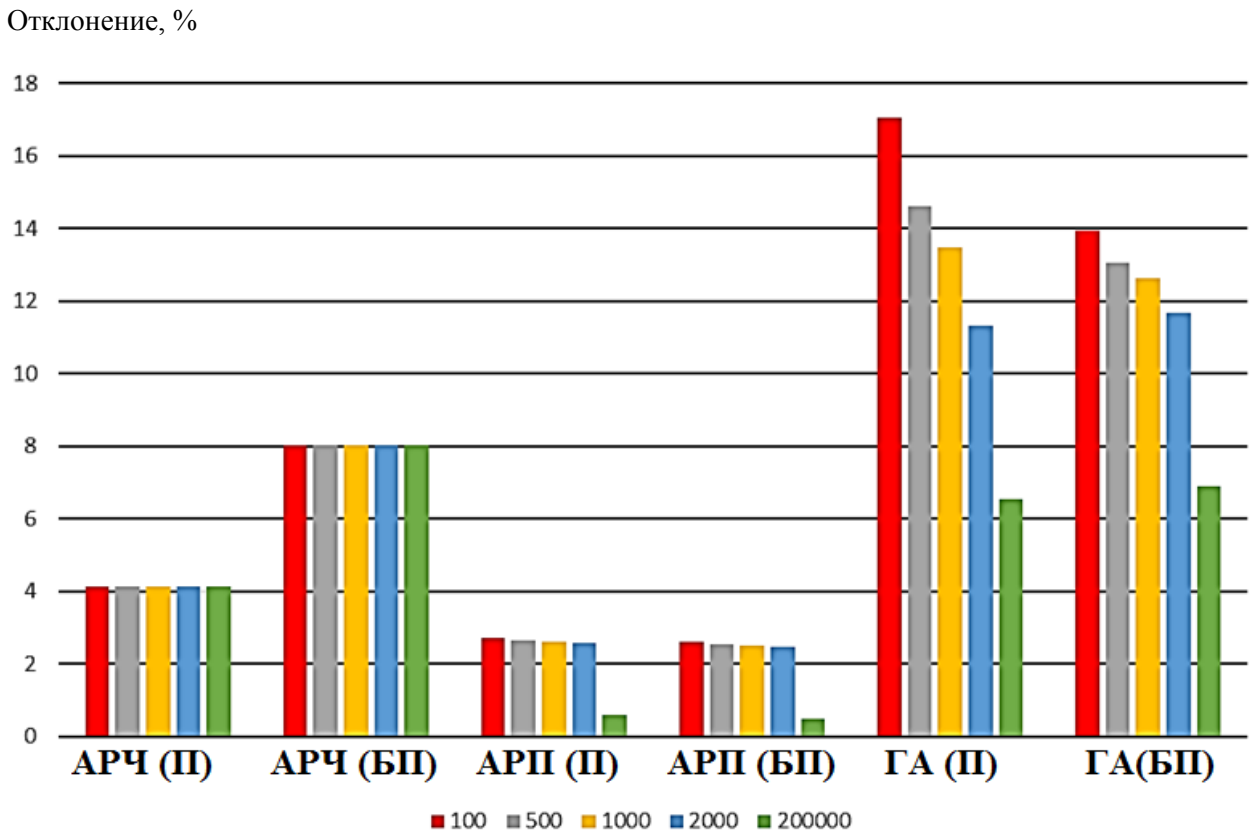



Рисунок 3.2 – Сравнение эффективностей алгоритмов, средних отклонений

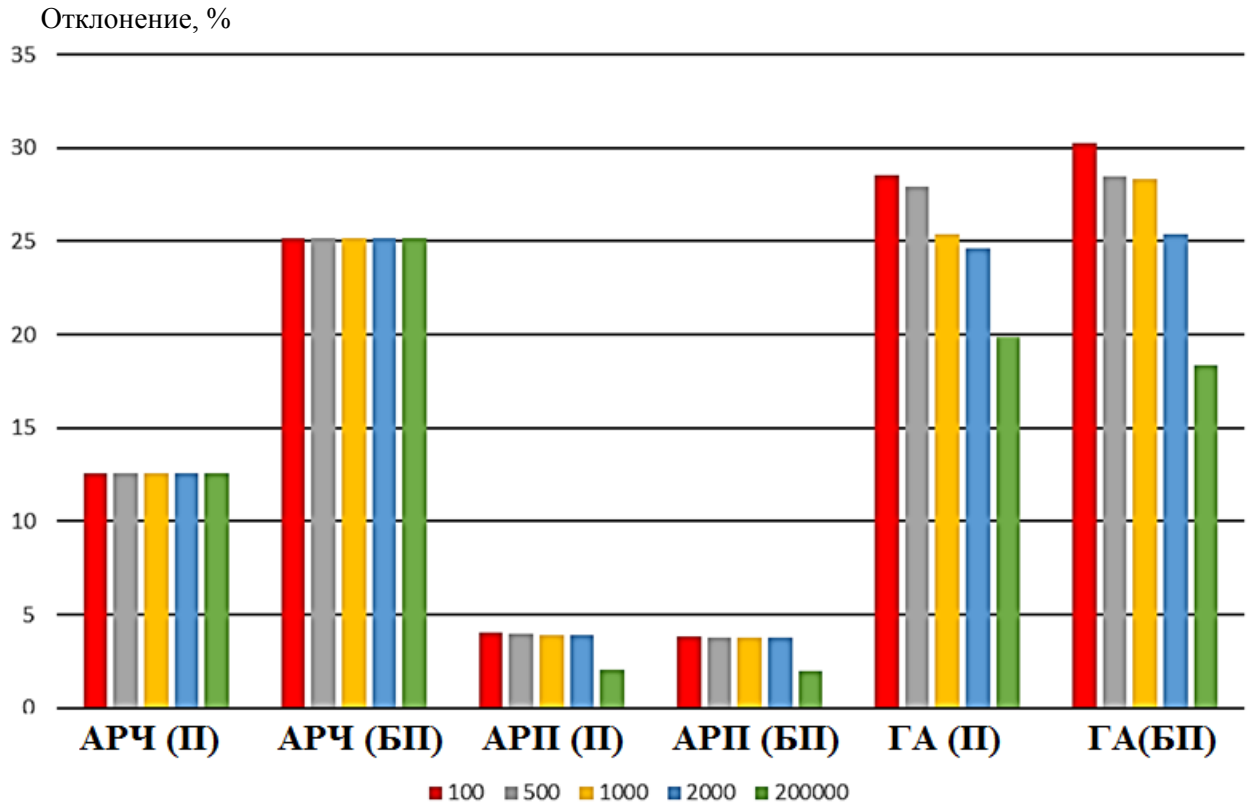



Рисунок 3.3 – Сравнение эффективностей алгоритмов, максимальных отклонений

Таблица 3.10 – Сравнение эффективностей алгоритмов, средних отклонений

| **Алг.** | **отклонение ΔP от наилучшего после заданного числа итераций, %** | | | | |
|---|---|---|---|---|---|
| | **100** | **500** | **1000** | **2000** | **200000** |
| АРЧ (П) | 4,132 | 4,127 | 4,125 | 4,123 | 4,118 |
| АРЧ (БП) | 8,025 | 8,024 | 8,024 | 8,024 | 8,024 |
| АРП (П) | 2,728 | 2,652 | 2,612 | 2,563 | 0,597 |
| АРП (БП) | 2,617 | 2,551 | 2,522 | 2,475 | 0,492 |
| ГА (П) | 17,057 | 14,606 | 13,486 | 11,323 | 6, 530 |
| ГА (БП) | 13,952 | 13,064 | 12,633 | 11,687 | 6,897 |

Таблица 3.11 – Сравнение эффективностей алгоритмов, максимальных отклонений

| **Алг.** | **отклонение ΔP от наилучшего после заданного числа итераций, %** | | | | |
|---|---|---|---|---|---|
| | **100** | **500** | **1000** | **2000** | **200000** |
| АРЧ (П) | 12,565 | 12,565 | 12,565 | 12,565 | 12,565 |
| АРЧ (БП) | 25,173 | 25,173 | 25,173 | 25,173 | 25,173 |
| АРП (П) | 4,072 | 3,982 | 3,9372 | 3,883 | 2,065 |
| АРП (БП) | 3,852 | 3,795 | 3,784 | 3,735 | 1,981 |
| ГА (П) | 28,558 | 27,896 | 25,411 | 24,643 | 19,893 |
| ГА (БП) | 30,289 | 28,442 | 28,309 | 25,363 | 18,361 |

В проведенных экспериментах были рассмотрены только последствия отказов КУ, их реальная частота и закон распределения отказов во времени не анализировались. Чтобы провести исследование, приближенное к реальным условиями, предполагается ввести моделирование отказов и восстановлений оборудования на основе статистических данных и добавить в модель изменения нагрузок в электроэнергетической системе в режиме реального времени.

### 3.1.7. Двухкритериальная задача оптимальной компенсации реактивной мощности

**Постановка задачи.** Оптимизационную задачу размещения КУ (3.1) можно не сводить к однокритериальной, а решать, учитывая оба критерия по-отдельности (поэтому штрафная функция тоже используется в обоих критериях):

$$\sum_{i=1}^{n} Q_i + G(Q) \to min$$

$$\Delta P(Q) + G(Q) \to min \quad (3.4)$$

Использование модели с двумя критериями вместо свертки критериев в один или перевода одного из критериев в разряд ограничений дает лицу, принимающему решения, намного больше информации о вариантах улучшения системы, кроме того, полученное множество альтернатив не зависит от изменения цен и расчетного периода, в то время как однокритериальную задачу пришлось бы решать для каждой комбинации цен и периода ($c_{cu}$, $c_p$ и $t$) в однокритериальной постановке задачи.

**Применение алгоритмов РИ**. Алгоритмы РИ могут быть адаптированы к решению многокритериальной задачи. Но по своей природе наиболее подходит для таких задач алгоритм роя светлячков, поскольку каждая частица выполняет очередной шаг, ориентируясь на все частицы, занимающие лучшие позиции. Это позволяет алгоритму лучше исследовать компромиссные области пространства поиска решений в многокритериальных задачах. На рисунке 3.4 показан пример. На четвертую частицу оказывают влияние все частицы, кроме второй и девятой. Влияние первой, шестой и восьмой частиц меньше, чем третьей, пятой и седьмой, так как расстояния между последними и четвертой частицей выше, а согласно формуле притяжение частиц, влияние убывает с ростом расстояния (см. подраздел 2.2.3):

$$v(X_i, X_k) = \begin{cases} \dfrac{\beta}{1 + \gamma r(X_i, X_k)}, \exists \theta\ f_\theta(X_i) < f_\theta(k) \\ 0, \nexists \theta\ f_\theta(X_i) < f_\theta(k) \end{cases}$$

где θ – индекс критерия, $r(X_{ij}, X_{kj})$ – декартово расстояние между частицами, β и γ эвристические коэффициенты алгоритма.

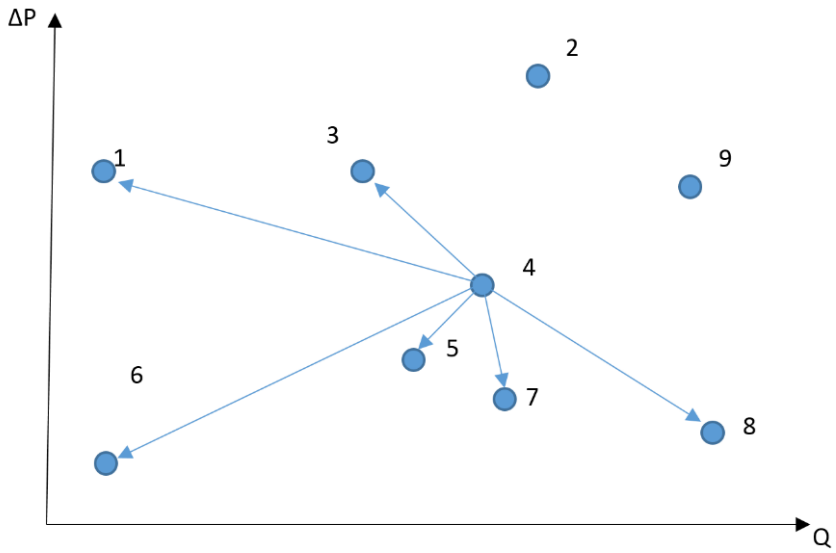



Рисунок 3.4 – Влияние двух критериев на частицу алгоритма роя светлячков

**Эксперимент.** Для эксперимента был рассмотрен фрагмент системы электроснабжения Таджикистана (рисунок 3.5). Номера узлов, которые рассматривались для установки КУ, обведены. Полученное адаптивным алгоритмом роя светлячков множество альтернатив показано в таблице 3.12 и на рисунке 3.6.

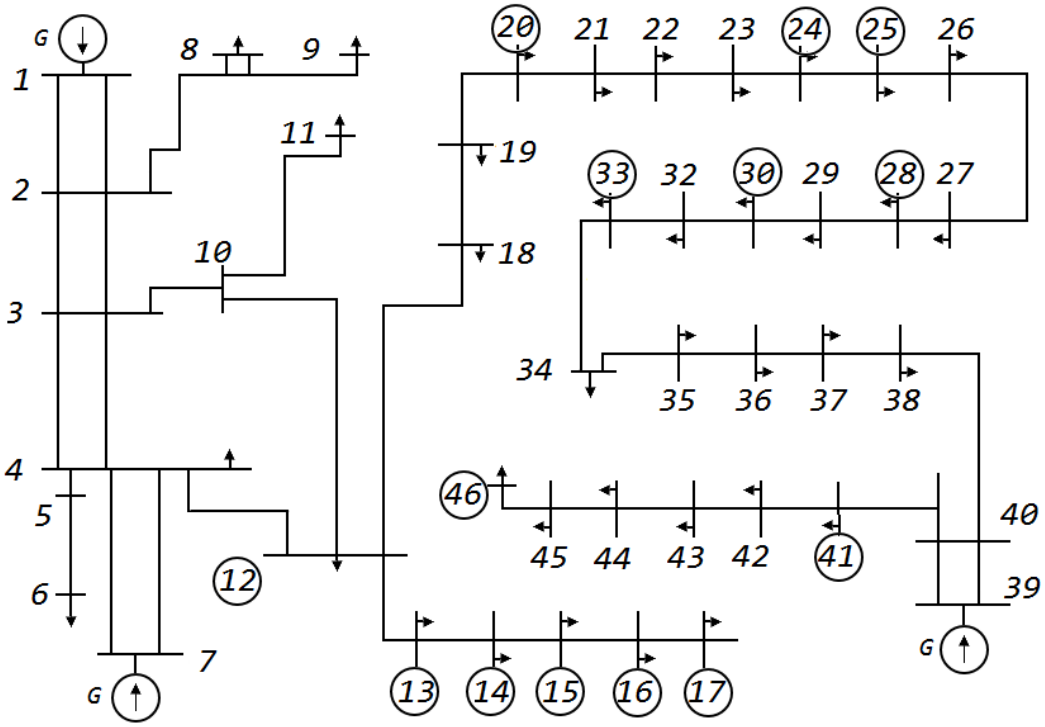



Рисунок 3.5 – Фрагмент электрической сети для двухкритериальной задачи оптимизации распределения КУ

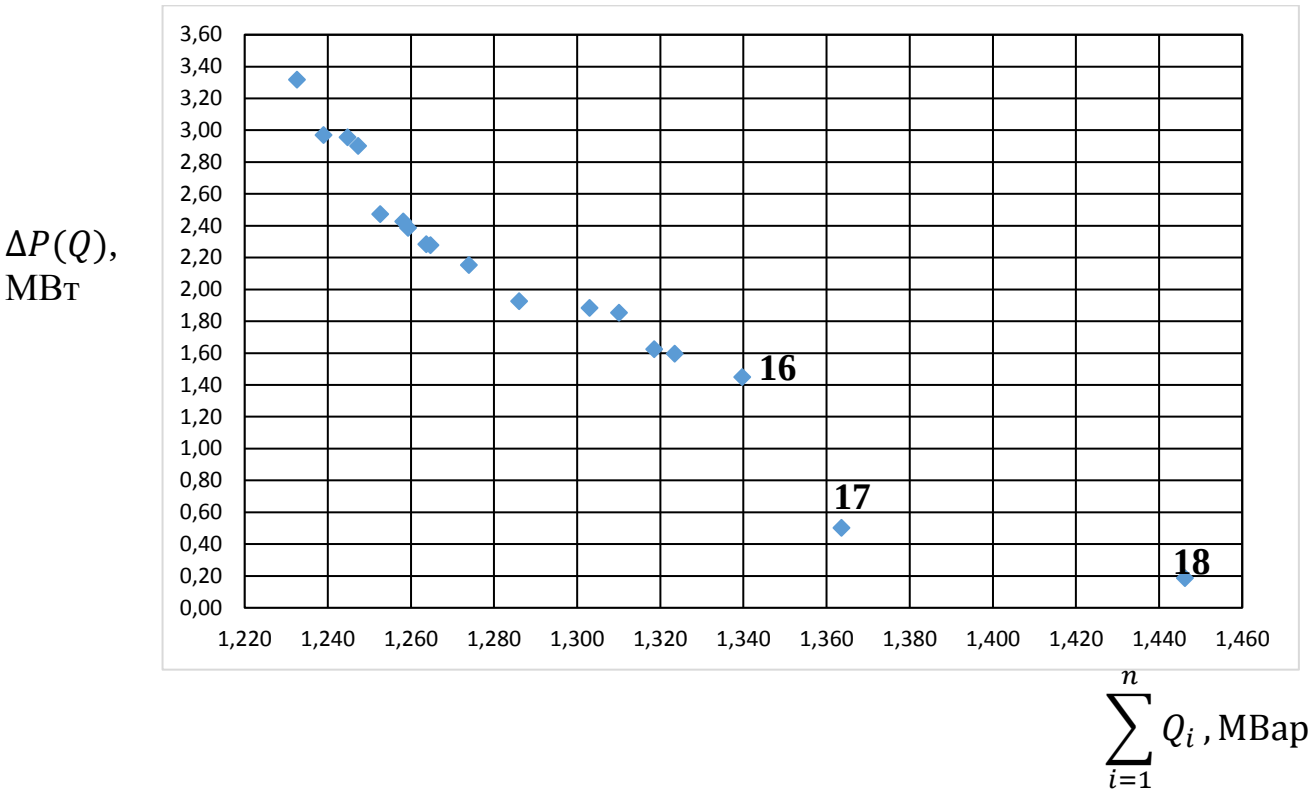



Рисунок 3.6 – Варианты решения двухкритериальной задачи

Таблица 3.12 – Варианты решения двухкритериальной задачи глубокой компенсации

| **Номер решения** | **Потери активной мощности, МВт** | **Мощность КУ, МВар** |
|---|---|---|
| 1 | 1,233 | 3,319 |
| 2 | 1,239 | 2,970 |
| 3 | 1,245 | 2,958 |
| 4 | 1,247 | 2,903 |
| 5 | 1,253 | 2,474 |
| 6 | 1,258 | 2,427 |
| 7 | 1,259 | 2,387 |
| 8 | 1,264 | 2,284 |
| 9 | 1,265 | 2,279 |
| 10 | 1,274 | 2,154 |
| 11 | 1,286 | 1,926 |
| 12 | 1,303 | 1,884 |
| 13 | 1,310 | 1,855 |
| 14 | 1,319 | 1,625 |
| 15 | 1,323 | 1,597 |
| 16 | 1,340 | 1,451 |
| 17 | 1,364 | 0,503 |
| 18 | 1,446 | 0,187 |

**Анализ результатов.** Во всех возможных парах решений из таблицы 3.12 одно решение лучше по потерям активной мощности, а второе лучше по затратам на КУ. Иными словами, в каждой паре одно решение лучше в краткосрочной перспективе, другое – в долгосрочной. Использование модели с двумя критериями вместо свертки критериев в один или перевода одного из критериев в разряд ограничений дает лицу, принимающему решения, намного больше информации о вариантах улучшения системы. Так можно заключить что существует некий предел, после которого эффективность компенсации реактивной мощности начинает снижаться: решение №18 лишь немного превосходит решения №17 по снижению потерь активной мощности, при этом существенно хуже с точки зрения вложений в КУ. Между решениями №16 и №17 значительное снижение потерь активной мощности требует намного меньших инвестиций. Особенно важно использовать многокритериальную оптимизацию, когда среди критериев часть влияет на краткосрочные показатели, а часть на долгосрочные. Это позволяет лицу, принимающему решение, соблюдать баланс тактических и стратегических целей.

## 3.2. Оптимизация коэффициентов трансформации

Помимо компенсации реактивной мощности эффективным способом повышения энергоэффективности является продольное регулирование напряжения путем регулирования коэффициентов трансформаторов под нагрузкой (РПН) [Сидоркин 1974; Жмак, 2002] для снижения потерь электроэнергии. В отличие от задачи компенсации реактивной мощности, данная задача является задачей дискретной оптимизации.

Регулирование напряжения в узлах электроэнергетической системы позволяет решать следующие основные задачи:

- обеспечение баланса выработки и потребления электрической энергии;
- обеспечение требуемого уровня напряжения на выводах потребителей электрической энергии.

Рассматриваемая задача является комбинаторной задачей оптимизации. Общее число всевозможных решений-комбинаций задачи можно определить, исходя из количества регулируемых трансформаторов ($n$) и количества возможных положений анцапф ($m$). Общее число комбинаций составляет $m^n$. Очевидно, что применить полный перебор всех возможных комбинаций невозможно на практике. Кроме того, данная комбинаторная задача относится к классу NP-трудных задача оптимизации. Поэтому время, требуемое для ее точного решения, независимо от метода оптимизации экспоненциально возрастает с увеличением числа трансформаторов.

В настоящее время задача выбора положения анцапф трансформаторов, как правило, решается с помощью разбиения задачи на отдельные части с оптимизацией каждой из них [Robbins, 2016; Quevedo, 2017], либо с использованием эвристических правил, часто основанных на нечеткой логике [Жмак, 2002; Spatti, 2010]. При этом для оптимизации систем электроснабжения целесообразно применять стохастические популяционные алгоритмы, которые способны находить удовлетворительные решения за приемлемое время и, главное, самостоятельно адаптироваться под условия решаемых задач и топологии оптимизируемых систем [Любченко, 2008; Keller, 2016; Manusov 2017].

### 3.2.1. Постановка задачи оптимизации коэффициентов трансформации

Задача оптимизации может быть определена как задача минимизации потерь активной мощности в сети путем настройки положений анцапф трансформаторов. При этом существуют дискретные ограничения, связанные с допустимыми положениями трансформаторов, и ограничение на соблюдение устойчивости режима работы системы:

$$\Delta P(Transf) \rightarrow \min \tag{3.5}$$

$$Transf = \{t_1, t_2, ,\ldots, t_n\} \tag{3.6}$$

$$1 \le t_i \le m,\ i = 1, \ldots, n \tag{3.7}$$

$$Stability(Transf) = 1. \tag{3.8}$$

Необходимо применить метод, который, во-первых, находил бы решение за приемлемое время, во-вторых, позволил бы искать решения оптимизационной задачи, как черного ящика. Второе требование связано с высокой сложностью интеграции оптимизационных методов в математические модели, используемые для расчетов режимов электросетей. Для его выполнения можно использовать комплекс «RastrWin», который позволяет выполнять расчет режима для заданных оптимизационным алгоритмом положений анцапф трансформаторов. Простым в реализации, быстрым в расчетах и при этом обеспечивающим некоторый уровень повышения энергоэффективности системы является направленный перебор, организованный следующим образом [Сидоркин, 1974; Манусов, 2017а].

1. Задать текущее положение анцапф всех регулируемых трансформаторов.

2. Пусть $k$ – номер рассматриваемого трансформатора, начать с $k = 1$.

3. Если $k \leq n$, то перейти к пункту 4. Иначе к пункту 6.

4. Выполнить расчет установившегося режима для всех 15 возможных положений анцапф трансформатора $k$. Положение, при котором потери активной мощности в линиях сети наименьшие, сохраняется в качестве текущего положения.

5. Установить $k = k + 1$. Перейти к пункту 3.

6. Если при выполнении пунктов 4 и 5 не было зафиксировано улучшение критерия, то завершить работу. Иначе перейти к пункту 2.

Алгоритм выполняет локальный поиск, перебирая все возможные положения анцапф каждого трансформатора по очереди. В случае, когда взаимное влияние трансформаторов мало, алгоритм может за 1–3 прохода по всем трансформаторам найти оптимальное решение или решение, близкое к оптимальному. Но в противном случае, эффективность метода может оказаться невысокой из-за попадания в одни и те же локальные экстремумы на каждой итерации. Этот недостаток присущ и другим эвристическим методам, оптимизирующим части системы по-отдельности [Robbins, 2016]. В работе [Жмак, 2002] показано, что данный алгоритм можно усовершенствовать, используя числа Фибоначчи, либо нечеткую логику. При этом снижается время работы алгоритма за счет сокращения перебираемых вариантов. Но качество получаемых решений остается таким же, как

и у направленного перебора. При этом для применения аппарата нечеткой логики сопряжено со значительными трудозатратами по составлению нечетких входных и выходных лингвистических переменных, и правил, описывающих выбор положений анцапф трансформаторов в той или иной ситуации [Spatti, 2010].

### 3.2.2. Применение адаптивных роевых алгоритмов для оптимизации коэффициентов трансформации

Для применения алгоритмов роевого интеллекта необходимо задать отображение положения частицы в пространстве поиска решений (вектор $X$) и варианта решения задачи (вектор $T$). Поскольку элементы вектора $X$ принимают значения в диапазоне от 0 до 1 (включая 0, но не 1), получим отображение:

$$t_i = \lfloor mx_i + 1 \rfloor, i = 1, \ldots, n. \quad (3.9)$$

Таким образом, положение анцапфы $i$-го трансформатора определяется как округленное в большую сторону произведение $x_i$ на число возможных положений анцапфы $i$-го трансформатора. Вычисление критерия $\Delta P(Transf)$ и проверка ограничения на устойчивость системы $Stability(Transf)$ выполняется на стороне модели решаемой задачи.

### 3.2.3. Вычислительные эксперименты в задаче оптимизации коэффициентов трансформации

Для экспериментов выбран фрагмент электроэнергетической системы [Манусов, 2017]. Данные по 17 наиболее важным трансформаторам энергосистемы приведены в таблице 3.13. В таблице 3.13 использованы следующие обозначения:

- Id – номер трансформатора;
- АТ – автотрансформатор;
- 3обм – трехобмоточный трансформатор;
- ВН – номер узла высокого напряжения;
- СН – номер узла среднего напряжения;
- НН – номер узла низкого напряжения.

Таблица 3.13 – Используемые трансформаторы

| № | Тип | ВН | СН | НН |
|---|---|---|---|---|
| 1 | АТ | 7 | 9 | 8 |
| 2 | АТ | 7 | 9 | 8 |
| 3 | АТ | 15 | 20 | 16 |
| 4 | АТ | 15 | 20 | 16 |
| 5 | АТ | 15 | 20 | 16 |
| 6 | АТ | 1 | 3 | 2 |
| 7 | АТ | 1 | 3 | 2 |
| 8 | АТ | 24 | 25 | 41 |
| 9 | АТ | 24 | 25 | 42 |
| 10 | 3обм | 4 | 6 | 5 |
| 11 | 3обм | 4 | 6 | 5 |
| 12 | 3обм | 21 | 23 | 22 |
| 13 | 3обм | 21 | 23 | 22 |
| 14 | 3обм | 11 | 18 | 45 |
| 15 | 3обм | 17 | 18 | 19 |
| 16 | 3обм | 17 | 18 | 19 |
| 17 | 3обм | 14 | 13 | 12 |

Все указанные трансформаторы имеют возможность регулирования напряжения посредством переключения РПН с 15 ступенями и шагом между коэффициентами 1.78. Сведения по узлам сети даны в таблице 3.14.

Результаты экспериментов для модели рассмотренной электросети приведены в таблице 3.15. В строках с обозначениями $Id_{анц}$ приведены рекомендованные разными алгоритмами положения анцапф. Строка «Потери, МВт» содержит потери активной мощности в линиях. Строка «Эффект, МВт» показывает, насколько потери снизились в результате оптимизации. Поскольку направленный перебор выполнялся в первый раз начиная с первого трансформатора, затем во второй с семнадцатого, результаты приведены в двух разных столбцах (НП 1, НП 17, соответственно). Для алгоритма роя частиц (PSO) указано число шагов алгоритма и значение варьируемого параметра, ограничивающего скорости частиц, β.

Таблица 3.14 – Параметры напряжения, активной и реактивной мощностей в установившемся режиме

| № | $U_{ном}$, кВ | $P$, МВт | $Q$, МВар | № | $U_{ном}$, кВ | $P$, МВт | $Q$, МВар |
|---|---|---|---|---|---|---|---|
| 1 | 230 | 179,5 | 134,6 | 25 | 115 | 0 | 0 |
| 2 | 11 | 28 | 19,6 | 30 | 11 | -70 | -46,1 |
| 3 | 115 | 163 | 117,4 | 31 | 11 | -60 | -39,5 |
| 4 | 115 | 0 | 0 | 32 | 230 | 0 | 0 |
| 5 | 11 | 0 | 0 | 33 | 230 | 0 | 0 |
| 6 | 38 | 18,5 | 12,9 | 34 | 230 | 0 | 0 |
| 7 | 230 | 109 | 76,3 | 35 | 230 | 0 | 0 |
| 8 | 11 | 0 | 0 | 36 | 230 | 0 | 0 |
| 9 | 115 | 0 | 0 | 37 | 230 | 0 | 0 |
| 10 | 230 | -459 | -302,7 | 38 | 230 | 0 | 0 |
| 11 | 230 | 0 | 0 | 39 | 230 | 0 | 0 |
| 12 | 7 | 4 | 2,4 | 40 | 230 | 0 | 0 |
| 13 | 38 | 12 | 8,4 | 41 | 11 | -70 | -24,5 |
| 14 | 230 | 0 | 0 | 42 | 11 | -70 | -46,1 |
| 15 | 230 | 0 | 0 | 43 | 230 | 0 | 0 |
| 16 | 11 | 14 | 9,8 | 45 | 11 | -60 | -29,9 |
| 17 | 230 | 0 | 0 | 44 | 230 | 0 | 0 |
| 18 | 38 | 45 | 30,1 | 46 | 230 | 0 | 0 |
| 19 | 11 | 15 | 9,3 | 47 | 230 | 0 | 0 |
| 20 | 115 | 124 | 86,8 | 48 | 115 | 0 | 0 |
| 21 | 115 | 0 | 0 | 49 | 115 | 0 | 0 |
| 22 | 11 | 8,4 | 6,3 | 50 | 115 | 0 | 0 |
| 23 | 38 | 20 | 13,6 | 51 | 115 | 0 | 0 |
| 24 | 230 | 0 | 0 | | | | |

### 3.2.4. Анализ результатов в задаче оптимизации коэффициентов трансформации

Алгоритм роя частиц показал более высокую эффективность, чем направленный перебор при использовании всего лишь 50 итераций. Нужно отметить, что эксперименты с 50 и 500 итерациями выполнялись как 2 различных запуска. Без ограничения максимальной скорости эффективность алгоритма роя существенно ниже, а выбор значения коэффициента β равным 0,3 дал наилучшие результаты. Можно сделать вывод, что без ограничения максимальной скорости (β = 1,0) частицы часто «пролетают» мимо окрестности наиболее низких

экстремумов, и даже попав в них, могут по инерции выйти обратно. При этом слишком сильно ограничение скорости (β = 0,1) ухудшает время поиска эффективного решения, отсюда значительная разница в решениях при 50 и 500 итерациях. При оптимизации с использованием метода роевого интеллекта в рассматриваемой сети можно добиться снижения потерь активной мощности с 48,01 МВт до 45,83 МВт, т.е. на 2,18 МВт или 4,5 %. Эта величина снижения потерь активной мощности обеспечивает экономию 19 миллионов рублей ежегодно.

В наилучшем решении среднее положение анцапф равно 9. Априори можно было предположить, что это значение будет близким к максимальному значению 16, так как потери активной мощности снижаются с ростом напряжения. Такое предположение может быть справедливым для радиальных сетей, в которых взаимные влияния сегментов сети слабые. Поэтому для оптимизации не радиальных сетей актуальность применение роевых алгоритмов особенно высока.

Таблица 3.15 – Результаты оптимизации коэффициентов трансформации

| **Алгоритм** | **Исходное решение** | **НП 1** | **НП 17** | **PSO 50 шагов, β = 0,1** | **PSO 500 шагов, β = 0,1** | **PSO 50 шагов, β = 0,3** | **PSO 500 шагов, β = 0,3** | **PSO 50 шагов, β = 1,0** | **PSO 500 шагов, β = 1,0** |
|---|---|---|---|---|---|---|---|---|---|
| Потери, МВт | 48,01 | 46,32 | 47,28 | 46,08 | **45,90** | **45,88** | **45,83** | 46,23 | 46,21 |
| Эффект, МВт | 0 | 1,69 | 0,73 | 1,93 | **2,11** | **2,13** | **2,18** | 1,78 | 1,80 |
| $Id_{анц}$ *1* | 2 | 11 | 2 | 9 | **10** | **10** | **8** | 11 | 10 |
| $Id_{анц}$ *2* | 2 | 6 | 2 | 9 | **8** | **10** | **8** | 8 | 8 |
| $Id_{анц}$ *3* | 6 | 10 | 6 | 11 | **12** | **14** | **10** | 12 | 11 |
| $Id_{анц}$ *4* | 6 | 7 | 11 | 8 | **11** | **14** | **10** | 12 | 16 |
| $Id_{анц}$ *5* | 6 | 10 | 10 | 10 | **12** | **14** | **10** | 12 | 10 |
| $Id_{анц}$ *6* | 6 | 6 | 6 | 9 | **10** | **9** | **8** | 7 | 5 |
| $Id_{анц}$ *7* | 6 | 6 | 6 | 8 | **9** | **8** | **7** | 8 | 16 |
| $Id_{анц}$ *8* | 6 | 7 | 10 | 10 | **10** | **12** | **10** | 10 | 11 |
| $Id_{анц}$ *9* | 6 | 10 | 7 | 10 | **11** | **8** | **10** | 7 | 11 |
| $Id_{анц}$ *10* | 2 | 2 | 2 | 6 | **10** | **4** | **9** | 9 | 8 |
| $Id_{анц}$ *11* | 2 | 2 | 2 | 9 | **10** | **4** | **7** | 9 | 10 |
| $Id_{анц}$ *12* | 6 | 6 | 6 | 8 | **7** | **8** | **9** | 2 | 10 |
| $Id_{анц}$ *13* | 6 | 6 | 6 | 10 | **7** | **8** | **8** | 2 | 10 |
| $Id_{анц}$ *14* | 6 | 10 | 6 | 11 | **12** | **6** | **10** | 7 | 10 |
| $Id_{анц}$ *15* | 6 | 7 | 4 | 6 | **10** | **4** | **6** | 8 | 16 |
| $Id_{анц}$ *16* | 6 | 6 | 5 | 7 | **10** | **4** | **16** | 7 | 8 |
| $Id_{анц}$ *17* | 6 | 12 | 12 | 8 | **8** | **11** | **4** | 2 | 16 |
| Средн. $Id_{анц}$ | 5 | 7 | 6 | 9 | **10** | **9** | **9** | 8 | 11 |

### 3.3. Оптимизация модели прогнозирования электропотребления

**Постановка задачи.** Важным направлением повышения эффективности функционирования электроэнергетических систем является прогнозирование энергопотребления. Задача важна для долгосрочного планирования и оперативного управления режимами функционирования электроэнергетических систем, определения необходимого количества и мощности генераторов, оценки экономичности, составления графиков выполнения ремонтных работ [Руденко, 2000; Hippert, 2001; Костин 2016]. Для этого на основании имеющихся наблюдений создается математическая модель графика нагрузки.

В данной работе рассматривалась модель, описывающая график нагрузки по месяцам как сумму постоянной составляющей, тренда и двух периодических составляющих. Модель записывается следующим образом:

$$P(t) = a_0 + a_1 + a_2\sin(w_2 t + \varphi_2) + a_3\sin(w_3 t + \varphi_3) \quad (3.10)$$

Качество модели определяется как:

$$\delta = \sum_{i=1}^{n} \frac{|P_i - P_i^*|}{P_i} 100\ \% \quad (3.11)$$

где $n$ – число месяцев в выборке

$P_i$ – известное по наблюдениям фактическое значение потребляемой мощности, МВт;

$P_i^*$ – полученное на основании модели (3.3) значение потребляемой мощности, МВт.

**Применение алгоритмов роевого интеллекта.** Для применения алгоритмов РИ достаточно определить правила перевода позиции частицы X в вектор коэффициентов модели

$Coeff = \{a_0, a_1, a_2, w_2, \varphi_2, a_3, w_3, \varphi_3\}$.

Для этого нужно задать диапазон возможных значений

$Coeff\_min = \{a_0^{min}, a_1^{min}, ..., \varphi_3^{min}\}$, $Coeff\_max = \{a_0^{max}, a_1^{max}, \ldots, \varphi_3^{max}\}$

В результате в векторной форме:

$Coeff = (Coeff\_max - Coeff\_min) \circ X + Coeff\_min$.

**Эксперимент.** Эксперименты проводились на среднемесячных значениях потребления Памира с 2007 по 2016 гг. по данным ОАО «Памир Энерджи» (Республики Таджикистан). Качество решений роевых алгоритмов сравнивалось с качеством решений, полученных ручным подбором параметров модели и полученных применением дискретного преобразования Фурье (ДПФ) с градиентным методом наискорейшего спуска. Последний метод заключается при применении ДПФ для получения начального приближения, поскольку ДПФ позволяет выделить постоянную составляющую сигнала и наиболее значимые периодические составляющие [Потапов, 2016]. Полученные коэффициенты модели (3.10) $a_0$, $a_2$, $w_2$, $\varphi_2$, $a_3$, $w_3$, $\varphi_3$ используются как начальное приближения для оптимизации алгоритмом наискорейшего спуска (коэффициент $a_0$ получает нулевое начальное значение, поскольку ДПФ не дает коэффициент тренда). Применение ДПФ с наискорейшим спуском наиболее сложно в реализации, но имеет наивысшую скорость работы. Ручной подбор коэффициентов требует наибольших затрат времени от человека, зато наиболее прост в реализации. Результаты применения различных методов приведены в таблице 3.16.

Таблица 3.16 – Результаты решения задачи оптимизации модели графика нагрузки

| **Коэффициенты модели (3.10) и ошибка (3.11)** | **Ручная настройка** | **ДПФ + градиент** | **АРИ** | **РИ** |
|---|---|---|---|---|
| $a_0$, МВт | 18,95 | 18,95 | 18,9 | 19,25 |
| $a_1$, МВт | 0,0043 | 0,0048 | 0,0047 | -0,0022 |
| $a_2$, МВт | 8,8 | 8,8 | 2,49 | 7,48 |
| $w_2$, Гц | 0,5237 | 0,5239 | 5,2382 | 6,8 |
| $\varphi_2$ | 1,009 | 1,009 | 2,135 | 0,61 |
| $a_3$, МВт | 2,59 | 2,59 | 8,59 | -0,11 |
| $w_3$, Гц | 1,047 | 1,047 | 5,76 | 4,57 |
| $\varphi_3$ | 0,785 | 0,785 | 2,172 | 0,25 |
| $\delta$ (3.3), % | 6,95 | 6,94 | 6,8 | 11,38 |

Затем уже настроенный алгоритм был применен для другой выборки данных, чтобы проверить устойчивость адаптации. Другие методы тоже были применены для этой выборки. Результаты эксперимента показаны в таблице 3.17. В строке

«Памир» еще раз указаны основные результаты первого эксперимента (с настройкой роевого алгоритма), в строке «ОЭС Сибири» показаны результаты второго эксперимента (с использование уже настроенного роевого алгоритма).

Таблица 3.17 – Сопоставление методов решения задачи оптимизации моделей прогнозирования графиков нагрузки

| **ЭЭС** | **Ручная настройка** | **ДПФ + градиент** | **АРИ** | **РИ** |
|---|---|---|---|---|
| Памир | 6,95 | 6,94 | 6,8 | 11,4 |
| ОЭС Сибири | 4,5 | 4,0 | 2,38 | 10,8 |

**Анализ результатов.** Адаптивные алгоритмы РИ позволяют быстро и точно подбирать численные параметры моделей в задачах регрессионного анализа. При этом применение роевых алгоритмов без настройки параметров дает результат быстро, но качество полученных моделей может быть ниже, чем при ручном подборе параметров или использовании дискретного преобразования Фурье с последующим улучшением методом градиентного спуска. Эксперименты подтвердили, что подобранные значения параметров дают улучшение эффективности не только на той задачи, для которой были подобраны, но и для других задач такого же класса.

## 3.4. Выводы по разделу 3

Раскрыты возможности применения адаптивных алгоритмов роевого интеллекта для решения задач оптимизации в проектировании и управлении электроэнергетическими системами.

Рассмотрена многокритериальная гибридная задача определения положений и мощностей компенсирующих установок в узлах систем электроснабжения при глубокой компенсации реактивной мощности. Задача имеет два критерия: снижение потерь активной мощности в системе при минимальных затратах на компенсирующие установки, пропорциональных их суммарной мощности. Задача имеет свойства как дискретной, так и непрерывной оптимизации, поскольку мощности установок непрерывны, но места их допустимого размещения

дискретны. Показано применение разработанных адаптивных алгоритмов роевого интеллекта для решения данной задачи и как однокритериальной, и как двухкритериальной. Для сведения двух критериев в один использован пересчет в сумму финансовых затрат на потери активной мощности и внедрение глубокой компенсации. Обосновано, что при решении задачи как многокритериальной, лицо, принимающее решения, получает намного больше информации о вариантах улучшения системы, особенно если среди критериев часть влияет на краткосрочные показатели, часть на долгосрочные и необходимо соблюдать баланс тактических и стратегических целей. Эксперименты проводились для различных электроэнергетических систем: подсистема электроснабжения Ангарского электролизного химического комбината, подсистема электроснабжения ОА «Уральский электрохимический комбинат», фрагмента энергетической системы Таджикистана. Применение адаптивных алгоритмов роевого интеллекта позволяет снизить потери активной мощности на 5–20 % при сроке окупаемости компенсирующих установок 3–5 лет. При этом существующие программные аналоги, применяемые для оптимизации электроэнергетических систем, не позволяют решать многокритериальные задачи или задавать все необходимые ограничения. Кроме того, проведено сравнение адаптивных алгоритмов роевого интеллекта с роевыми алгоритмами без разработанной адаптации и другими оптимизационными методами (генетический алгоритм, градиентные методы). Эффективность решений адаптивных алгоритмов выше на 2–9 %, чем роевых алгоритмов без адаптации.

Важным направлением исследований в области методов оптимизации является решение задач, условия которых динамически меняются в процессе решения. Экспериментально было показано, что в задачах динамической оптимизации, которые часто возникают в оперативном управлении, важно понимать особенности работы алгоритмов роевого интеллекта. Теоретически и экспериментально было показано, что наилучшим для таких задач является алгоритм роя пчел. Поскольку он в каждый момент работы гарантированно хранит несколько удаленных друг от друга решений. В отличие от большинства других

алгоритмов роевого интеллекта, которые с течением времени сходятся в окрестности одного экстремума. Однако возможно использование указанных свойств алгоритма роя пчел для переноса в другие роевые алгоритмы, с целью создания алгоритма, обладающего свойствами, наиболее ценными для особенностей решаемой задачи.

Адаптивные алгоритмы роевого интеллекта использованы в решении задачи выбора коэффициентов трансформации в системах электроснабжения 110–220 кВ. Экспериментально показано, что при оптимизации с использованием адаптивных алгоритмов роевого интеллекта в рассмотренной системе можно добиться снижения потерь активной мощности с 48,01 МВт до 45,83 МВт (2,18 МВт). Выявлено, что рассматриваемый класс задач обладает сложной топологией, при которой оказываются малоэффективными часто применяемые в данной задаче методы, основанные на направленном переборе, эвристических правилах или нечеткой логике. Указанные методы дают высокую эффективность только для радиальных сетей, в которых взаимные влияния различных сегментов сети слабые. Показано, что эффект от применения алгоритмов роевого интеллекта на 28,9 % выше, чем эффект от других методов для рассмотренной электроэнергетической системы.

# 4. Исследование адаптивных алгоритмов роевого интеллекта в задаче календарного планирования

Раздел 4 посвящен применению адаптивных алгоритмов роевого интеллекта в решении задачи календарного планирования (КП). Задача КП выбрана в качестве области сравнения разработанных алгоритмов с другими методами по нескольким причинам. Во-первых, задача является NP-трудной задачей оптимизации и обладает теми же особенностями и сложностями, что и задачи в области электроэнергетики. Во-вторых, задаче КП посвящено множество исследований, и она часто применяется как тестовая для сопоставления эффективности различных методов решения задач комбинаторной оптимизации. В-третьих, рассмотренная задача является актуальной для многих производственных и образовательных организаций. Раздел состоит из пяти подразделов, раскрывающих содержательное описание задачи КП (подраздел 4.1), математическую модель задачи КП класса job-shop (4.2), краткий обзор применяемых методов решения (4.3), описание предложенного подхода к применению алгоритмов РИ (4.4), описание проведенных экспериментов и их результатов (4.5).

## 4.1. Содержательное описание задачи календарного планирования

Во всех сферах человеческой деятельности необходимо составлять расписания и планы. Увеличение трудоемкости составления планов и развитие математического аппарата привели к созданию отдельного направления в области исследования операций – теории расписаний. Важность задач планирования начала стремительно возрастать с XX века с развитием конвейерного производства. Систематические исследования в этой области начались в середине XX века, Термин «теория расписаний» введен Р. Беллманом в 1956 году [Лазарев, 2011]. Важность задач планирования до начала XX века была невысокой, но начала стремительно увеличиваться с появлением поточного конвейерного производства Генри Форда, который предложил разбивать работы на небольшие упорядоченные

операции, и работами Генри Ганта над повышением эффективности руководства на промышленных предприятиях.

Задачи теории расписаний относятся к классу задач комбинаторной оптимизации или упорядочивания. Активное исследование и разработка теории расписаний начались с 50-х годов XX века. Одним из главных вопросов теории расписаний в то время стала классификация задач и установление их сложности [Лазарев, 2011]. Обзоры по задачам теории расписаний представлены в работах Гэри и Джонсона, [Johnson, 1954], Лоуера [Lawler, 2993], Танаева [Танаев, 1989], Брукера и др. [Brucker, 2006].

Можно использовать такое определение: «Теория расписаний – это раздел исследования операций, в котором строятся и анализируются математические модели КП (т.е. упорядочивания во времени) различных целенаправленных действий с учетом целевой функции и различных ограничений» [Лазарев, 2011].

Задачи КП подразумевают планирование во времени, то есть составление указаний, что именно в какие моменты времени необходимо выполнять. Такие указания в общем случае называются расписанием. Для одной и той же ситуации может быть составлено множество различных расписаний. Очевидно, что расписания отличаются друг от друга по множеству показателей, поэтому возникает задача формирования критериев оценки расписаний и выбора наиболее удовлетворяющего эти критериями расписания. Задача КП характеризуется целью, моделью объекта управления, набором требований (работ), которые необходимо выполнить для достижения цели, критериями оценки. Ключевыми понятиями в теории расписаний являются «работа» («требование») и «ресурс» (исполнитель»).

Работы состоят между собой в различных отношениях, одним из наиболее важных является отношение предшествования между работами. Отношение предшествования задает ограничения на последовательности выполнения работ. Наиболее частой причиной таких зависимостей являются технологические ограничения, часто начало одних работ зависит от результатов других (подготовка станка к обработке изделия должна предшествовать обработке), хотя возможны и другие ограничения. В задачах КП помимо отношений между работами

используется множество разнообразных ограничений: крайние сроки выполнения отдельных работ и совокупностей работ, ограничения по ресурсам, сроки, раньше которых нельзя начинать определенные работы, ограничения, связанные с объектом управления (ограничения трудового законодательства в системах, где используется человеческий ресурс, различные правила и требования в системах массового обслуживания, в образовательных учреждениях и т.д.).

Определив типы работ, необходимые ресурсы по каждой из них, отношения между работами, наличие ресурсов, ограничения и построив модель выполнения работ с использованием ресурсов, можно с помощью некоторого алгоритма составить допустимое расписание, если такое существует, или показать, что составление расписания для указанных условий невозможно. Содержательно задачи планирования являются оптимизационными, т.е. заключаются в выборе среди множества допустимых вариантов (расписаний, допускаемых условиями задачи) тех, которые обеспечивали бы достижение наилучшего (оптимального) значения критерия. Под оптимальностью понимается время выполнения плана, затраты ресурсов, прибыль и прочие показатели.

## 4.2. Математическая модель оптимизационной задачи job-shop

Как показано в исследовании [Танаев, 1989], большинство задач КП связаны с многостадийными обслуживающими системами, под которыми понимаются системы, в которых процесс выполнения требования разделен на несколько стадий (операций). При существующем разнообразии производственных систем для большинства из них в качестве основы для математической модели может быть использована постановка задачи КП из работы [Танаев, 1989]. Она заключается в следующем. Даны конечное множество требований $N = \{1, 2, ..., n\}$ и конечное множество приборов $M = \{1, 2, ..., m\}$. В качестве требований могут выступать работы, заказы, партии деталей и т.п. В качестве требований – станки, бригады, исполнители. Для выполнения $i$-го требования необходимо выполнить определенную последовательность стадий. Стадии $q$ ($1 \le q \le R_i$) требования $i$

сопоставляется определенное подмножество приборов $M^i_q$. Прибор может обслуживать только одно требование в один момент времени. Каждое требование $i \in$ характеризуется известной строго соблюдаемой последовательностью обслуживания приборами:

$$L^i = (l_1{}^i, l_2{}^i, \ldots, l_{ri}{}^i).$$

Временная длительности обслуживания стадии $q$ требования $i$ известны ($t_{iq}$) для всех стадий всех требований. Искомый календарный план является совокупностью $\{s_1(t), s_2(t), \ldots, s_m(t)\}$ кусочно-постоянных непрерывных слева функций, каждая из которых задана на интервале времени $0 \le t < \infty$ и принимает целочисленные значения от 0 до $N$

$$s=\{s_1(t), s_2(t), \ldots, s_m(t)\}$$

Если $s_l(t') = i$ , $s_l(t') \neq 0$ , $l \in M$ , $i \in N$ , то в момент времени $t'$ прибор $l$ обслуживает требование $i$. Если $s_l(t') = 0$, то в этом момент прибор не обслуживает ни одно из требований. При этом учитываются следующие ограничения:

- все требования готовы к выполнению в начальный момент времени $(t = t_0)$;

- ни одна из стадий не может быть прервана после начала выполнения;

- прибор может начинать выполнение стадии сразу же, как только он освобождается и выполнены предшествующие стадии данного требования.

Из последнего пункта следует, что достаточным, но не необходимым признаком оптимального плана является полное отсутствие простоев приборов. Нужно учесть, что нахождение не оптимального, а лишь удовлетворяющего всем требованиям решения уже представляет из себя непростую алгоритмическую задачу. Расчет качества (критерия оптимальности) расписания осуществляется исходя из времени завершения всех требований. В частных случаях возможны и другие критерии.

Сложности задачи календарного планирования видны уже на этапе ее постановки. Основной сложностью является необходимость алгоритмического построения расписаний, так как в общем случае невозможно сформулировать модель задачи и критерий оптимальности в виде системы алгебраических

выражений. Большинство задач теории расписаний относятся к NP-трудным, по оценкам [Rinnooy, 1975], их доля составляет 80 % от общего числа задач.

Среди задач КП в многостадийных системах наиболее сложными являются задачи оптимизации систем класса «job-shop», в данной работе рассматриваются именно они. Основным отличием от задач класса «open-shop» является заданный порядок стадий в каждом требовании, а от задач класса «flow-shop» – различия в порядке обслуживания разных требований. Задача КП класса «job-shop» относится к NP-трудным, если приборов больше двух [Лазарев, 2011; Танаев, 1989]. Как показано в [Gromicho, 2012], число комбинаций для задачи «job-shop» с $n$ требованиями и $m$ приборами (требование содержит $m$ стадий) прямо пропорционально величине $(n!)^m$. В таблице 4.1 приводятся данные, иллюстрирующие возрастание времени решения задач КП полным перебором в зависимости от ее размерности, при условии, что оценка одного решения занимает лишь $10^{-10}$ секунд [Puchta, 2004].

Таблица 4.1 – Время решения задачи КП в зависимости от размерности

| **Размерность задачи (одинаковое количество требований, стадий и приборов)** | **Время решения полным перебором** |
|---|---|
| 25 | 0,0034 секунды |
| 50 | 31 час |
| 75 | $1{,}2 \cdot 10^5$ лет |
| 100 | $4 \cdot 10^{12}$ лет |

## 4.3. Обзор используемых методов оптимизации

### 4.3.1. Детерминированные методы в задачах календарного планирования

**Переборные методы.** Наиболее простым методом решения комбинаторных задач является полный перебор, но, как следует из данных таблицы 4.1, даже используя очень мощные вычислительные системы, задачи с десятками требований и приборов за приемлемое время решить методом полного перебора невозможно. Объем вычислений можно сократить, используя методы направленного перебора.

Во многих работах предлагается рассматривать задачи КП как задачи целочисленного программирования [Танаев, 1989]. Но применение методов целочисленного программирования эффективно лишь для задач «job-shop» малой размерности или же при двух приборах (в частных случаях при трех), в остальных случаях затраты времени слишком велики.

Для определенного подкласса задач КП может быть применен метод ветвей и границ [Brucker, 1994]. Каждому узлу дерева поиска, то есть подмножеству допустимых частичных планов, сопоставляется оценка величины критерия. Минимальная оценка для терминальных узлов дерева (полностью построенных планов) принимается в качестве нижней границы оптимального плана. Оценка нижней границы необходима, чтобы отбрасывать подмножества частично построенных планов, заведомо не содержащих оптимальных решений задачи, этот прием сокращает размерность поискового пространства. Достоинствами метода являются универсальность, гарантированное получение наилучшего решения, высокая скорость работы для задач малой и средней размерности. Но применение метода ветвей и границ в задачах КП затруднено в случаях высокой размерности из-за огромных затрат времени и памяти для хранения промежуточных результатов, кроме того, возникает сложность с оценкой составляемого плана по частичному решению задачи.

**Градиентные методы.** Применение градиентных методов в задачах КП также затруднено, так как задачи не являются дифференцируемыми, имеют высокую степень «овражности» и множество локальных экстремумов. Следовательно, эти задачи обладают всеми свойствами, критически снижающими эффективность градиентных методов даже с использованием возвратов из-за попаданий в локальные экстремумы и невозможности аналитически определить производную критерия оптимальности.

**Эвристические методы.** Примеры жадных эвристических правил для задач КП даны в [Секаев, 2005], например: «выбирать на каждом шаге этап, выполнение которого завершится как можно раньше». Более сложные алгоритмы основаны на расстановке приоритетов требованиям в зависимости от различных факторов или

на комбинировании правил. Так как способов назначения приоритетов и вариаций эвристических правил бесконечно много, то существует большое число модификаций метода.

Даже очень сложные эвристические правила могут оказаться неэффективными [Кочетов 2005; Кормен, 2005; Скиена, 2013]. Так в работе [Кочетов, 2005] описаны жадные эвристические алгоритмы для решения задачи КП, при этом указано, что эвристики могут давать большое отклонение от оптимума и для повышения эффективности необходимо дополнять их трудоемкой процедурой локального поиска. Локальный поиск применяется и в методе, названном авторами «Shifting Bottleneck Procedure» [Adams, 1991], особенностью метода для рассматриваемой задачи является итерационный возврат к ранее составленным расписаниями для их переоптимизации. Также применяются алгоритмы, комбинирующие различные эвристики и локальные правила [Fisher, 1963; Норенков, 1999; Секаев, 2005].

### 4.3.2. Стохастические методы в задачах календарного планирования

Наиболее эффективными в решении задач КП являются эвристические недетерминированные методы, в которых косвенно учитывается опыт предыдущих решений. Как правило, такие методы можно описать цепями Маркова и доказать их сходимость к глобальному оптимуму задачи [Кочетов, 2000]. Но при этом время сходимости может быть очень велико, поэтому на практике ставится задача найти не глобальный оптимум, а достаточно близкое к нему решение.

Для решения задач КП успешно применятся как популяционные, так и не популяционные стохастические методы. К последним относятся поиск с запретами, имитация отжига. К недостаткам методов относится необходимость тщательно настраивать и адаптировать их. Очень часто в КП применяют эволюционные методы, особенно генетический алгоритм [Норенков, 1999; Секаев, 2009; Harrabi, 2017; Anshulika, 2017]. Достоинством алгоритма является простота реализации, к недостаткам относятся невысокая эффективность на задачах, обладающих сложной

топологией пространства решений и высокой размерностью; необходимость многократных вычислений критерия для низкоэффективных решений, и часто слишком долгое время выполнения алгоритма, так как «операции пересечения и мутации обычно не используют структуры данных, специфичных для данной задачи, вследствие чего большинство переходов дает низкокачественные решения и процесс поиска наилучшего решения протекает медленно» [Скиена, 2013]. Помимо этого, трудно эффективно записать необходимые для однозначного построения расписания данные в структуре хромосомы так, чтобы операции мутации и кроссинговера не приводили к слишком сильным искажениям расписаний [Секаев, 2005].

## 4.4. Применение роевых алгоритмов для решения задачи календарного планирования

Как показано во втором разделе, для применения алгоритмов РИ к решению оптимизационных задач необходимо вносить изменения не в модель оптимизируемой системы и не в алгоритм РИ, а в процесс взаимодействия между ними. Рассматриваемая задача КП является комбинаторной, а среди алгоритмов РИ большая часть ищет решения в непрерывном пространстве поиска (алгоритмы роя частиц, роя пчел, алгоритм светлячков), либо в пространстве поиска, заданном графом (алгоритм колонии муравьев). Для этих двух видов необходимо определить два соответствующих способа применения к задаче КП, для которой сама задача представления решения является нетривиальной.

### 4.4.1. Алгоритмы с непрерывным пространством поиска

При решении задачи КП алгоритмами РИ с непрерывным пространством поиска, дискретное расписание необходимо некоторым образом закодировать в форме вектора непрерывных управляющих переменных (позиции частицы). Можно поступить следующим образом: записать расписание как вектор длиной $k$ целых чисел:

$$k = \sum_{i=1}^{n} r_i \ , \qquad (4.1)$$

где $n$ – количество требований, а $r_i$ – количество этапов $i$-го требования.

При этом состояние частицы в пространстве решений является вектором $k$ целых чисел, каждое из которых соответствует индексу требования. Но в такой способ представления решения имеет недостаток: поскольку $i$-е требование не может входить в решение более $r_i$ раз, возникает необходимость проверки этого условия при каждом перемещении частицы роя и корректировки решения в случае невыполнения условия.

В данной работе применен более эффективный способ представления решения. Номера требований определяются по следующей схеме: каждые $r_i$ элементов вектора соответствуют $i$-му требованию, $i$=1, 2, ...$n$. Затем вектор сортируется по вещественным значениям элементов. В результате получается некоторая последовательность индексов требований, в зависимости от значений элементов вектора управляющих переменных, с которым и работает алгоритм роя частиц. Например, пусть вектор, характеризующий положение частицы в задаче с двумя двухэтапными требованиями, имеет вид {0,93; 0,21; 0,72; 0,48}. Ставим каждому элементу вектора в соответствие индексы требований: {(0,63; 1), (0,21; 1), (0,72; 2), (0,48:2)}. После сортировки получаем {(0,21; 1), (0,48:2), (0,72; 2), (0,93; 1)}, что соответствует следующему порядку выполнения: 1-й этап первого требования, 1-й этап второго требования, 2-й этап второго требования, 2-й этап первого требования.

Таким образом, процесс декодирования можно записать следующим образом.

1. Записать вектор этапов всех требований, согласно (4.1)

$L = \{ l_1{}^1, l_2{}^1, ..., l_{rn}{}^n \}$.

2. Присоединить каждому этапу позицию частицы ($X$).

$LX = \{ (x_1 l_1{}^1), (x_2 l_2{}^1), \ldots, (x_k l_{rn}{}^n) \}$,

где число частиц $k$ соответствует числу всех этапов (4.1).

3. Отсортировать вектор $LX$ по убыванию $X$.

4. Удалить элементы $x$ их элементов вектор $LX$.

5. Полученный вектор $L^*$ еще содержит искомый порядок приоритетов этапов, поскольку в нем не соблюдается последовательность этапов внутри отдельного требования.

6. Для каждого требования провести замену элементов вектора $L^*$ на номера этапов этого требования в их заданном порядке.

Приведем пример. Рассмотрим 3 требования по 3 операции:

$l_1 = \{ l_1^{\,1}, l_2^{\,1}, l_3^{\,1}\}$, $l_2 = \{ l_1^{\,2}, l_2^{\,2}, l_3^{\,2}\}$, $l_3 = \{ l_1^{\,3}, l_2^{\,3}, l_3^{\,3}\}$.

Пусть вектор $X = \{0{,}11; 0{,}72; 0{,}57; 0{,}92; 0{,}43; 0{,}21; 0{,}12; 0{,}40; 0{,}33\}$.

В этом случае после сортировки

$LX = \{\{l_1^{\,2}; 0{,}92\}, \{l_2^{\,1}; 0{,}72\}, \{l_3^{\,1}; 0{,}57\}, \{l_2^{\,2}; 0{,}43\}, \{l_2^{\,3}; 0{,}4\}, \{l_3^{\,3}; 0{,}33\}, \{l_3^{\,2}; 0{,}21\}, \{l_1^{\,3}; 0{,}12\}, \{ l_1^{1}; 0{,}11\}\}$.

Окончательно искомый вектор $L^{**} = \{ l_1^{\,2}, l_1^{\,1}, l_2^{\,1}, l_2^{\,2}, l_1^{\,3}, l_2^{\,3}, l_3^{\,2}, l_3^{\,3}, l_3^{\,1}\}$.

Вектор $L^{**}$ показывает не порядок начал выполнения этапов, а порядок назначения этапам позиции во временном графике обработки с минимальным возможным временем начала.

Процесс декодирования никак не затрагивает алгоритм роя частиц, его можно считать частью блока вычисления критерия оптимальности. Таким образом обеспечивается высокая гибкость алгоритма: добавление новых ограничений и прочие модификации математической модели задачи КП не приведут к необходимости менять структуру алгоритма роя частиц.

### 4.4.2. Алгоритмы с дискретным пространством поиска

Как описано в подразделе 2.2.2, алгоритм колонии муравьев решает задачу оптимизации, выполняя многократный обход некоторого графа. Поэтому для решения задачи КП алгоритмом колонии муравьев необходимо представить создание решения задачи в форме обхода ориентированного графа. В данной работе предложен следующий подход.

Для формирования расписания необходимо на каждом шаге определить, очередную стадию какого требования выбрать для наискорейшего размещения на нужном приборе. В этом случае у графа будет $k + 1$ вершин, $k$ определяется выражением (4.1). Первая вершина графа соединена направленными ребрами только со второй вершиной, вторая вершина соединена с третьей и так далее. Направленные ребра соответствуют требованиям. Частица двигается по графу, выбирая на каждом шаге ребро – требование. Если требование попало в маршрут столько раз, сколько у него стадий, то соответствующие ребра будут далее игнорироваться частицей.

Пояснение представлено на рисунках 4.1 и 4.2. Пусть даны три требования: *A, B, C*. Требование *A* состоит из двух стадий, требования *B* и *C* состоят каждое из трех. Число *k* будет равно сумме 2 + 3 + 3 = 8. Исходный граф показан на рисунке 4.1.

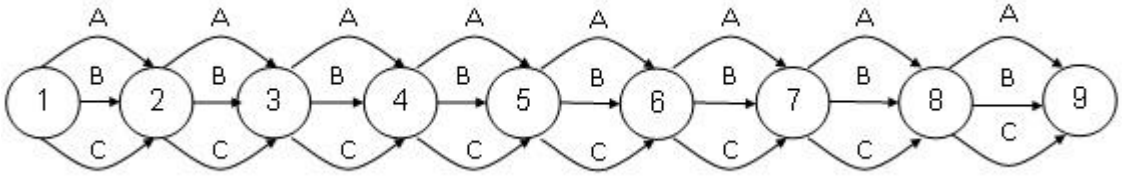



Рисунок 4.1 – Пример графа

Допустим, что частица на первом шаге выбрала требование *A*, затем *B* и опять *A*. Требование *A* содержит две стадии, поэтому оставшиеся ребра, обозначенные на рисунке буквой *A*, будут игнорироваться частицей на следующих шагах. Пусть на них частица выбирает последовательно *C, C, B, C*, тогда в узле графа №8 остается допустимым только ребро *B*. В результате приведенного прохода будет получена последовательность *A, B, A, C, C, B, C, B*. Из него легко получить вектор последовательности выбора стадий:

$L^{**} = \{l_1{}^A, l_1{}^B, l_2{}^A, l_1{}^C, l_2{}^C, l_2{}^B, l_3{}^C, l_3{}^B\}$.

Маршрут частицы по графу для данного примера показан на рисунке 4.2. Выбранные ребра показаны более толстыми линиями, пунктиром показаны ребра, которые игнорировались частицей исходя из сделанных выборов.

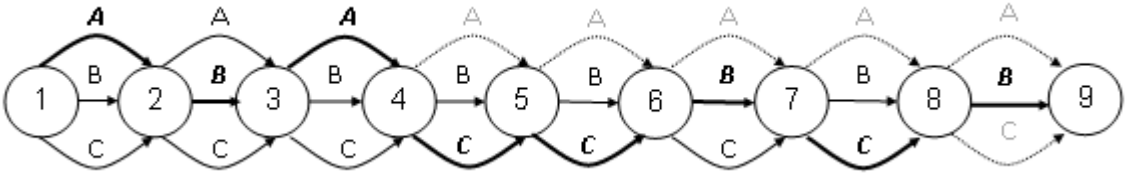



Рисунок 4.2 – Пример прохода частицы по графу

Рассматриваемая задача отличается от задачи обхода взвешенного неориентированного графа, на основании которой приводилось описание алгоритма колонии муравьев. Однако, для адаптации алгоритма к этим условиям не требуется никаких модификаций алгоритма, все необходимые настройки в предложенном в данной работе описании уже есть. Достаточно в список вершин графа, допустимых для начала обхода поместить самую левую вершину. Граф не является взвешенным, это эквивалентно единичному весу всех дуг графа. Это равнозначно нулевому значению коэффициента α алгоритма в формуле 2.3 (см. подраздел 2.2.2).

## 4.5. Вычислительные эксперименты на тестовых задачах

### 4.5.1. Описание набора тестовых задач

Эксперименты проводились с помощью описанного в пунктах 3 и 5 программного обеспечения и на самостоятельно разработанных задачах КП, задачах из библиотеки Operational Research Library [Beasley] (размерности задач указаны в приложении Г), реальной производственной задачи обработки станочных плит [Секаев, 2005]. Наилучшие известные решения задач из Operational Research Library были взяты из [Pezzella, 2000]. Пример задачи la01 приведен в таблице 4.2. Например, требование «a» состоит из 5 этапов, вначале должно выполняться на приборе «B» 21 единицу времени, потом на приборе «A» 53 единицы времени и т.д.

Таблица 4.2 – Пример исходных данных задачи job-shop (la01)

| **Требование** | **Прибор** | **Время** | **Прибор** | **Время** | **Прибор** | **Время** | **Прибор** | **Время** | **Прибор** | **Время** |
|---|---|---|---|---|---|---|---|---|---|---|
| a | B | 21 | A | 53 | E | 95 | D | 55 | C | 34 |
| b | A | 21 | D | 52 | E | 16 | C | 26 | B | 71 |
| c | D | 39 | E | 98 | B | 42 | C | 31 | A | 12 |
| d | B | 77 | A | 55 | E | 79 | C | 66 | D | 77 |
| e | A | 83 | D | 34 | C | 64 | B | 19 | E | 37 |
| f | B | 54 | C | 43 | E | 79 | A | 92 | D | 62 |
| g | D | 69 | E | 77 | B | 87 | C | 87 | A | 93 |
| h | C | 38 | A | 60 | B | 41 | D | 24 | E | 83 |
| i | D | 17 | B | 49 | E | 25 | A | 44 | C | 98 |
| j | E | 77 | D | 79 | C | 43 | B | 75 | A | 96 |

Так как алгоритмы РИ являются стохастическими, полученные результаты зависят от случайных факторов, при реализации алгоритма на компьютере таким фактором является последовательность псевдослучайных чисел, которые используются в вычислениях. Потому одного запуска алгоритма недостаточно, чтобы правильно оценить эффективность используемых коэффициентов, поэтому часто алгоритм запускают несколько раз, чтобы выбрать лучшее решение. Для минимизации влияния случайных факторов необходимо выполнить процедуру многократного запуска и выбора лучшего решения несколько раз и затем взять средний и наилучший результаты, которые и покажут эффективность решения задачи на некотором наборе коэффициентов.

### 4.5.2. Влияние эвристических коэффициентов роевых алгоритмов

**Алгоритм роя частиц**. Для проведения эксперимента было произвольно выбрано несколько известных тестовых задач (la01, la15, la17, la21) [Beck, 2005] и задача обработки станочных плит [Секаев, 2005] (3plity). Чтобы получить значительную дисперсию полученных при решении задач результатов, использовались относительно небольшие количества частиц и итераций (20 частиц и 100 итераций).

Было сгенерировано 500 наборов случайных коэффициентов $<\alpha_1, \alpha_2, \omega, v_{max}>$. В данном эксперименте по каждому набору коэффициентов задача КП решалась

100 раз. Для каждой *i*-й задачи и *j*-го набора коэффициентов было получено 50 000 кортежей вида $<\alpha^j_1, \alpha^j_2, \omega^j, v^j_{max}, \varphi^{ij}_k>$, где $\varphi^{ij}_k$ обозначает качество решения *i*-й задачи на *k*-м запуске, полученного на наборе $<\alpha^j_1, \alpha^j_2, \omega^j, v^j_{max}>$. Затем выбирались лучшие результаты по каждым десяти запускам $\varphi^{ij}_k$ и вычислялось их среднее значение:

$$\varphi^{*ij} = \frac{1}{10}\sum_{h=1}^{10} \min_{10(h-1)<k\leq 10h}\left(\varphi^{ij}_k\right)$$

Величина $\varphi^{*ij}$ была принята за показатель эффективности *j*-го набора коэффициентов для *i*-й задачи. В итоге, по каждой задаче было сформировано 500 кортежей вида $<\alpha^j_1, \alpha^j_2, \omega^j, \beta^j, \varphi^{*ij}>$, с которыми и проводились исследования.

Анализ следует начать с построения гистограмм распределений результатов решения по каждой из задач. На рисунках 4.3–4.7 показаны соответствующие гистограммы для тестовых задач. Для задач с наименьшей размерностью (la01 и 3plity) на большей части наборов коэффициентов были получены решения, которые предположительно являются наилучшими возможными решениями данных задач. При этом значительное количество наборов коэффициентов дали результаты хуже.

Для задачи наибольшей размерности (la21) доля наборов коэффициентов, давших решения, близкие к наилучшему известному решению, составила всего 1,4 %, что указывает на наличие небольшого числа параметров, которые дают наиболее эффективные решение. Очевидно, что для получения таких параметров перебором или случайным поиском необходимо было бы сгенерировать и проверить очень большое количество вариантов, что неприемлемо при решении реальных производственных задач. Задачи la15 и la17 по размерностям меньше la21 и больше la01 и 3plity, соответственно и результаты по ним получены с долями наилучших решений меньшими, чем в la01 и 3plity, но большими, чем в la21.

Из проведенного анализа видно, что полученные решения задач КП отличаются на разных наборах коэффициентов и чем выше размерность задачи, тем меньше эффективных наборов при прочих равных условиях.

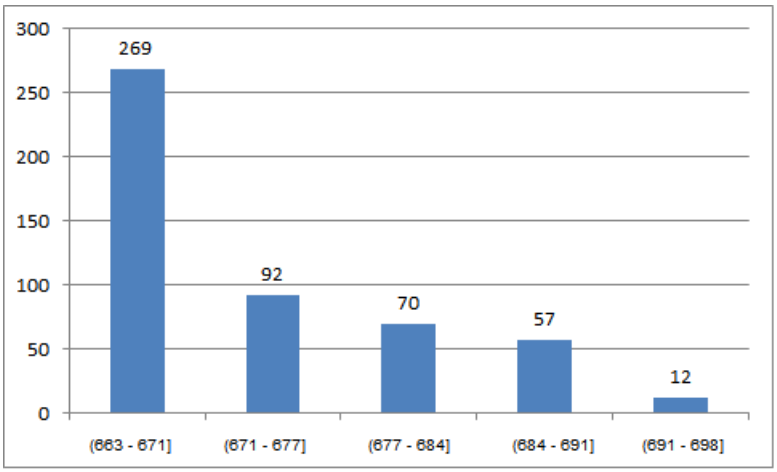



Рисунок 4.3 – Распределение решений задачи la01

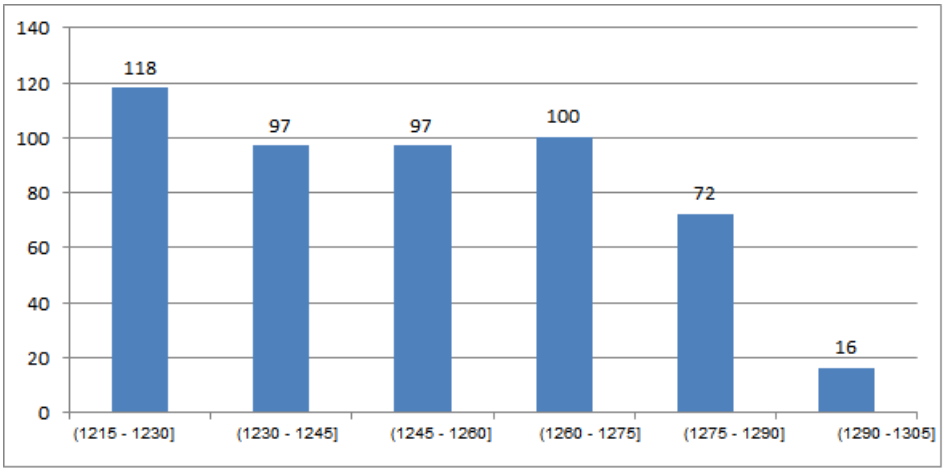



Рисунок 4.4 – Распределение решений задачи la15

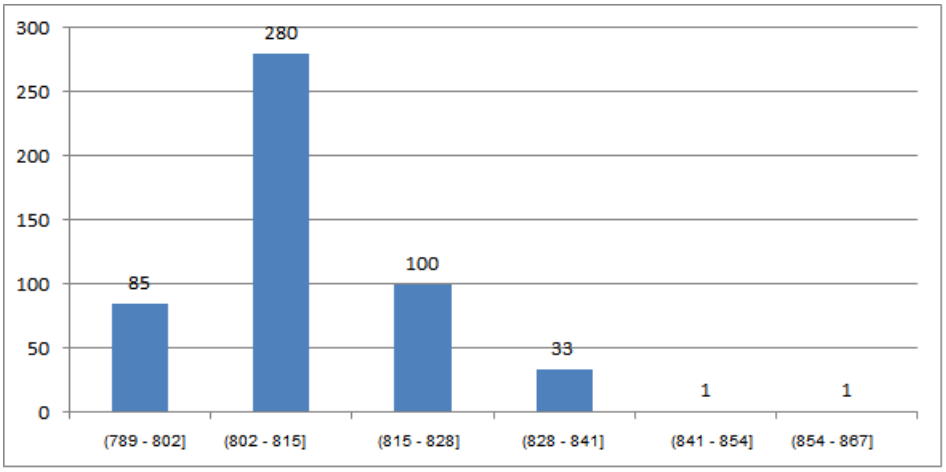



Рисунок 4.5 – Распределение решений задачи la17

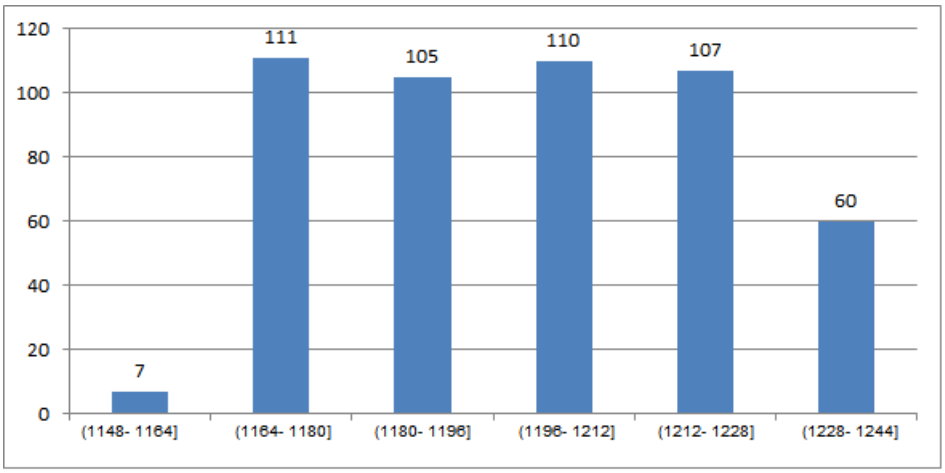



Рисунок 4.6 – Распределение решений задачи la21

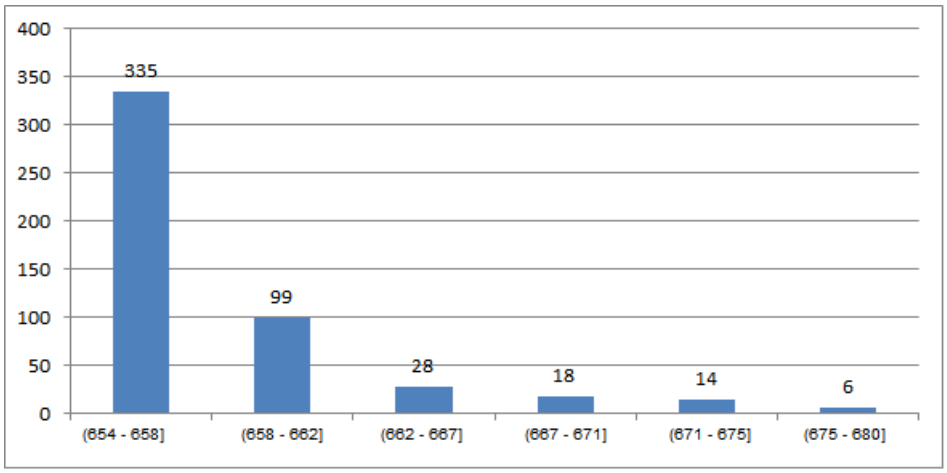



Рисунок 4.7 - Распределение решений задачи 3plity

**Алгоритм колонии муравьев.** В данном пункте для иллюстрации влияния значений коэффициентов на эффективность алгоритмов роя частиц и колонии муравьев приводятся некоторые результаты экспериментальных исследований. В таблице 4.3 приведены значения коэффициентов и параметров колонии муравьев, которые были отобраны с помощью эволюционной адаптации как наилучшие для некоторых задач.

В таблице 4.3 введены следующие обозначения:

$I_z$ – количество запусков, по которым проводилось усреднение (с одинаковыми коэффициентами);

$L_{min}$ – лучшее полученное решение;

$L_{avg}$ – усредненное по $I_z$ запускам решение;

$I_{mk}$ – количество итераций алгоритма колонии муравьев;

$C_{ant}$ – количество муравьев;

β – коэффициент влияния феромона на выбор дуги графа;

γ – линейный коэффициент нанесения феромона;

λ – коэффициент учета лучшего текущего решения;

μ – нелинейный коэффициент нанесения феромона;

ρ – коэффициент испаряемости феромона.

Из полученных данных следует, что лучшие найденные наборы коэффициентов различаются даже при решении одной и той же задачи, поиск взаимосвязи коэффициентов, их корреляции является одним из направлений дальнейшего развития работы.

Таблица 4.3 – Примеры лучших наборов коэффициентов, полученных при использовании адаптивного алгоритма колонии муравьев

| **Задача** | $I_z$ | $L_{min}$ | $L_{avg}$ | $I_{mk}$ | $C_{ant}$ | **α** | **β** | **ρ** | **γ** | **λ** |
|---|---|---|---|---|---|---|---|---|---|---|
| abz6 | 40 | 948 | 977,333 | 1000 | 100 | 0,63 | 2,0 | 0,696 | 28 | 1,3 |
| abz6 | 40 | 945 | 982,524 | 1000 | 100 | 1,1 | 1,0 | 0,499 | 900 | 1,3 |
| ft10 | 40 | 950 | 995,866 | 1000 | 100 | 0,63 | 1,2 | 0,7 | 1000 | 1,1 |
| ft10 | 20 | 951 | 1006,1 | 1000 | 50 | 0,3781 | 1,711 | 0,97 | 1108,7 | 2,277 |
| la10 | 20 | 784 | 805,4 | 1000 | 100 | 0,392 | 2,7701 | 0,2998 | 425,5387 | 1,9615 |
| la17 | 20 | 785 | 798,25 | 1000 | 100 | 0,0341 | 1,7968 | 0,5461 | 949,1535 | 4,5589 |
| la15 | 20 | 1207 | 1215,45 | 1000 | 100 | 0,531 | 1,72 | 0,663 | 1032 | 1,2 |
| 3plity | 10 | 657,55 | 662,075 | 30 | 10 | 0,2782 | 0,4251 | 0,4919 | 296,61 | 2,5373 |
| 3plity | 10 | 657,55 | 662,3275 | 30 | 10 | 0,1754 | 0,5705 | 0,3836 | 326,05 | 2,5083 |
| la01 | 10 | 666 | 670,2 | 200 | 20 | 0,2262 | 0,8665 | 0,6883 | 903,3459 | 2,0363 |
| la01 | 10 | 666 | 669,4 | 200 | 20 | 0,3542 | 0,6527 | 0,8001 | 120,8012 | 2,1305 |
| la21 | 10 | 1107 | 1150,9 | 2000 | 100 | 0,7478 | 1,1134 | 0,3488 | 790,57 | 2,2236 |

### 4.5.3. Сравнение результатов использованных методов

Эксперименты показали существенное улучшение результатов по сравнению с полученными ранее (без эволюционного подбора коэффициентов), что отражено в таблице 4.4 для алгоритма роя частиц и 4.5 для алгоритма колонии муравьев. Выделены столбцы с лучшими результатами до применения адаптации и после. В тестовых задачах считается, что все длительности заданы в условных временных единицах, но задача «3plity» является реальной, в которой все длительности

измерялись в часах. Поэтому для определенности будет полагать, что все указанные в таблицах результаты показывают длительности полученных планов в часах.

Таблица 4.4 – Повышение эффективности при адаптации алгоритма роя частиц

| **Задача** | **Результат без адаптации** | | **Результат с адаптацией** | **Число частиц** | **Число итераций** |
|---|---|---|---|---|---|
| | средний | лучший | | | |
| abz6 | 1003,6 | 951 | 943 | 100 | 100 |
| ft06 | 59,77 | 55 | 55 | 10 | 5 |
| ft10 | 992,1 | 937 | 937 | 300 | 300 |
| la01 | 690,97 | 666 | 666 | 100 | 20 |
| la10 | 963,47 | 958 | 958 | 10 | 5 |
| la15 | 1267,63 | 1217 | 1207 | 50 | 50 |
| la17 | 815,1 | 794 | 784 | 100 | 100 |
| la21 | 1143,17 | 1091 | 1085 | 500 | 500 |
| 3plity | 681,24 | 659,5 | 657,55 | 50 | 10 |

Таблица 4.5 – Повышение эффективности при адаптации алгоритма колонии муравьев

| **Задача** | **Результат без адаптации** | | **Результат с адаптацией** | **Число частиц** | **Число итераций** |
|---|---|---|---|---|---|
| | средний | лучший | | | |
| abz6 | 1005,3 | 980 | 945 | 1000 | 100 |
| ft06 | 55,16 | 55 | 55 | 30 | 10 |
| ft10 | 1038,8 | 1017 | 950 | 1000 | 100 |
| la01 | 673,08 | 666 | 666 | 30 | 10 |
| la10 | 958 | 958 | 958 | 1000 | 100 |
| la15 | 1220,6 | 1211 | 1207 | 1000 | 100 |
| la17 | 809,32 | 796 | 784 | 1000 | 100 |
| la21 | 1168,1 | 1121 | 1107 | 1000 | 100 |
| 3plity | 664,782 | 657,55 | 657,55 | 30 | 10 |

Из таблиц 4.4 и 4.5 видно, что планы, полученные с адаптацией, значительно эффективнее не только средних, но даже и лучших результатов, полученных при ручном подборе коэффициентов. Повышение эффективности можно объяснить тем, что подбор параметров снижает риск сходимости процесса поиска решения к локальным неэффективным экстремумам. Кроме повышения качества решений,

значительно снижаются и трудозатраты на их получение, так как отпадает необходимость подбирать коэффициенты вручную. Это особенно важно для задач исследования и сравнения различных методов. Например, получить для задачи «abz6» результат лучше 980 часов методом колонии муравьев авторам не удавалось очень долгое время, пока не была выдвинута идея об адаптации, после ее реализации сразу был получен результат 945 часов, отличающийся от наилучших [Pezzella, 2000] на 0,2 %.

### 4.5.4 Сравнение с результатами других авторов

Как сказано выше, в работе [Pezzella, 2000] даны наилучшие известные решения тестовых задач Operational Research Library, по задаче обработки плит результаты приводятся в [Плотников, 2011] и [Секаев, 2009]. Эти решения можно считать наилучшими решениями, которые можно получить за приемлемое время. В таблице 4.6 приводится сравнение результатов адаптивного алгоритма колонии муравьев с указанными наилучшими результатами. По многим тестовым задачам полученные решения совпали с наилучшими известными, по большинству задач отклонение не превысило 1 %. Наибольшее отклонение составляет 5,63 %. Учитывая погрешности, которые всегда возникают во время планирования и выполнения плана такие результаты можно считать очень близкими. В таблице 4.7 проводится аналогичное сравнение для разработанного адаптивного алгоритма роя частиц.

Для адаптивного алгоритма роя частиц также по многим тестовым задачам решения совпали с наилучшими известными, и по большинству задач отклонение не превысило 1 %, наибольшее отклонение составляет 2,38 %. Таким образом, адаптивный алгоритм роя частиц показал результаты ближе к наилучшим известным, чем адаптивный алгоритм колонии муравьев. Можно предположить, что это связано с меньшей зависимостью алгоритма роя частиц от структуры задачи, в то время как для алгоритма колонии муравьев необходимо представлять задачу в виде графа, что не всегда можно сделать эффективно для работы

алгоритма. Возможно, построение графа задачи КП для решения алгоритмом колонии муравьев должно быть проведено с большим учетом особенностей задачи. Однако построение графа задачи КП наилучшим образом не входит в задачи данной работы, направленной, в первую очередь, на повышение эффективности алгоритмов роя частиц. Очевидно, что более эффективная реализация алгоритма колонии муравьев или другого алгоритма РИ, специально настроенная на решения именно задач КП, точно так же может быть улучшена путем адаптивного подбора коэффициентов с помощью предложенного в данной работе подхода.

Также по некоторым тестовым задачам было проведено сравнение с результатами генетического алгоритма комбинирования эвристик и алгоритма роя частиц, специально настроенного под решение задач КП [Плотников, 2011]. Результаты сравнения приведены в таблице 4.7. Из таблицы 4.7 видно, что адаптивный алгоритм роя частиц показал наилучший результат по всем задачам из тестовой выборки, адаптивный алгоритм колонии муравьев проиграл только ему. Причем в работе [Плотников, 2011] приводятся наилучшие результаты алгоритмов по десяти запускам, а не по одному, как для алгоритмов, разработанных в данной работе.

Таблица 4.6 – Сравнение результатов адаптивных алгоритмов с наилучшими известными

| **Задача** | **Адаптивный алгоритм роя частиц** | **Адаптивный алгоритм колонии муравьев** | **Лучший известный результат** | **Относительное отклонение от лучших, %** |
|---|---|---|---|---|
| abz6 | 943 | 945 | 943 | 0 |
| ft06 | 55 | 55 | 55 | 0 |
| ft10 | 937 | 950 | 930 | 0,753 |
| la01 | 666 | 666 | 666 | 0 |
| la10 | 958 | 958 | 958 | 0 |
| la15 | 1207 | 1207 | 1207 | 0 |
| la17 | 784 | 784 | 784 | 0 |
| la21 | 1073 | 1107 | 1048 | 2,386 |
| 3plity | 657,55 | 657,55 | 657,55 | 0 |

К недостаткам разработанных адаптивных алгоритмов по сравнению алгоритмами, которые использовались в сравнении, можно отнести более высокую скорость работы. Поэтому если требуется получить решение не обязательно очень близкое к оптимальному, но максимально быстро, адаптивные алгоритмы РИ применять нецелесообразно. Однако для решения большинства задач планирования имеет смысл потратить больше времени на планирование, чтобы получить план как можно лучше. Вместе с тем развитие облачных технологий, позволяющих получать значительные вычислительные ресурсы при необходимости, и эффективность распараллеливания разработанных адаптивных алгоритмов позволяет считать скорость работы менее важным критерием, чем качество полученных решений.

Таблица 4.7 – Сравнение адаптивных алгоритмов роя частиц (АРЧ) и колонии муравьев (АКМ) с алгоритмами комбинирования эвристик (КЭ) и роя частиц (РЧ)

| **Задача** | **АРЧ** | **АКМ** | **КЭ** | **РЧ** |
|---|---|---|---|---|
| ft06 | 55 | 55 | 55 | 55 |
| ft10 | 943 | 945 | 985 | 961 |
| la01 | 666 | 666 | 666 | 666 |
| la15 | 1207 | 1207 | 1263 | 1207 |
| la21 | 1073 | 1107 | 1129 | 1128 |

### 4.5.5. Устойчивость эволюционной адаптации

Устойчивость эволюционной адаптации была проверена следующим образом:

- были случайным образом выбраны три задачи КП из набора задач Lawrence [Beck, 2000]: LA02, LA18 и LA20;

- адаптивный алгоритм роя частиц был с помощью эволюционной адаптации настроен на указанные три задачи, по описанной во втором разделе процедуре;

- настроенный алгоритм был запущен для других задач выборки;

- для сравнения результатов были выбраны рекомендуемые значения коэффициентов алгоритма роя частиц из работ [Eberhart, 2001; Pedersen, 2010b].

Таким образом, проверялось, будет ли адаптация, проведенная только на трех задачах выбранного класса, эффективной при решении других задач этого же класса.

Рекомендуемые параметры из [Eberhart, 2001]:

$P_{Eberhart}$: $\alpha_1 = 1{,}49445$, $\alpha_2 = 1{,}49445$, $\omega = 0.729$.

В работе [Pedersen, 2010b] приводятся различные наборы значений коэффициентов в зависимости от размерности задачи и числа итераций алгоритма. Для решаемых задач ближе всего следующие значения:

$P_{Pedersen}$: $\alpha_1 = -0{,}2746$, $\alpha_2 = 4{,}8976$, $\omega = -0{,}3488$.

Обе указанные работы не рассматривают ограничения скорости $\beta$ как настраиваемый эвристический коэффициент, в них скорость частицы ограничена только диапазоном допустимых значений оптимизируемой переменной. В предложенной в данной работе схеме алгоритма роя частиц это соответствует значению коэффициента $\beta$, равному единице.

Во всех экспериментах использовалось небольшое число частиц и итераций (50 частиц и 100 итераций), поскольку в рассматриваемых задачах планирования вычисление критерия (составление плана) занимает существенное время. Кроме того, ограничение на число частиц и итераций позволяет лучше проявиться влиянию коэффициентов.

Адаптивный алгоритм роя частиц использовал 50 хромосом и 100 итераций. Количество усредняющих запусков роевого алгоритма для каждого набора параметров было выбрано равным десяти. В результате были получены следующие значения коэффициентов:

$P_{apso}$: $\alpha_1 = 1{,}76428$, $\alpha_2 = 1{,}38203$, $\omega = 0{,}730135$, $\beta = 0{,}280868$.

Таблица 4.8 показывает результаты экспериментов, лучшие по десяти запускам для каждой задачи. Кроме того, приводятся наилучшие известные результаты.

Все варианты показали результаты, близкие к наилучшим. При этом, допустив, что наилучшие известные решения являются оптимальными, получим, что использование коэффициентов из работы [Eberhart, 2001] дало оптимальное

решение по 16 задачам из 21 (76 %), из работы [Pedersen, 2010b] по 11 (57 %), и коэффициенты, найденные эволюционной адаптаций, по 17 (81 %).

Таблица 4.8 – Усредненные результаты с использованием различных значений коэффициентов

| **Задача** | $P_{Eberhart}$ | $P_{Pedersen}$ | $P_{apso}$ | **Best known** |
|---|---|---|---|---|
| LA01 | 666 | 670 | 666 | 666 |
| LA02 | 658 | 663 | 658 | 655 |
| LA03 | 597 | 622 | 597 | 597 |
| LA04 | 590 | 594 | 590 | 590 |
| LA05 | 593 | 593 | 593 | 593 |
| LA06 | 926 | 926 | 926 | 926 |
| LA07 | 890 | 890 | 890 | 890 |
| LA08 | 863 | 863 | 863 | 863 |
| LA09 | 951 | 951 | 951 | 951 |
| LA10 | 958 | 958 | 958 | 958 |
| LA11 | 1222 | 1222 | 1222 | 1222 |
| LA12 | 1039 | 1039 | 1039 | 1039 |
| LA13 | 1150 | 1150 | 1150 | 1150 |
| LA14 | 1292 | 1292 | 1292 | 1292 |
| LA15 | 1207 | 1207 | 1207 | 1207 |
| LA16 | 979 | 982 | 956 | 945 |
| LA17 | 784 | 804 | 784 | 784 |
| LA18 | 859 | 896 | 859 | 848 |
| LA19 | 866 | 894 | 865 | 842 |
| LA20 | 911 | 950 | 907 | 902 |
| LA21 | 1123 | 1188 | 1128 | 1046 |
| **Сумма** | **19124** | **19354** | **19101** | **18966** |
| **Суммарное отклонение** | **158** | **388** | **135** | **0** |
| **Средний квадрат отклонения** | **7,52** | **18,48** | **6,43** | **0** |

Таким образом, показано, что настройка коэффициентов может быть выполнена по ограниченному числу задач некоторого класса для повышения эффективность роевого алгоритма на других задачах этого класса. Иными словами, адаптация является устойчивой к вариации условий.

Наибольшее время решения одного экземпляра составляло около 0,5 секунды с использованием процессора Intel Core i7 с тактовой частотой 2,4 ГГц. Минимальное время составляло 0,08 секунды. Такой разброс объясняется разным

размером задач. Время вычисления не зависит от значений коэффициентов. Время настройки путем эволюционной адаптации составляло около 2 часов при распараллеливании на 4 ядра. Предложенный алгоритм адаптации эффективно распараллеливается, поскольку это происходит на уровне генетического алгоритма, как было показано во втором разделе.

## 4.6. Выводы по разделу 4

Проанализированы содержательная и математическая постановки задач календарного планирования многостадийных обслуживающих систем. Для исследования эффективности разработанных алгоритмов и сравнения с результатами других авторов использовались NP-трудные задачи календарного планирования класса job-shop, преимущественно на широко распространенных задачах из библиотеки Operation Research Library.

В данных задачах само представление решения в компактной численной форме и восстановление расписания по нему являются нетривиальными задачами, затрудняющими применение стохастических популяционных алгоритмов. Поэтому были разработаны два специальных интерфейса к задаче для применения роевых алгоритмов, выполняющих поиск в непрерывном пространстве поиска решений (как алгоритм роя частиц) и в дискретном (как алгоритм колонии муравьев). При этом сохранен основной принцип применения роевых алгоритмов, выдвинутый в работе, который заключается в максимальном разделении алгоритма и решаемой задачи оптимизации. Таким образом, каждый из алгоритмов роевого интеллекта может быть применен по одной из двух предложенных схем. На тестовых задачах Operational Research Library было определено, что алгоритмы роевого интеллекта могут быть успешно применены для решения задач календарного планирования.

Проведены исследования влияния коэффициентов роевых алгоритмов на качество получаемых календарных планов, а также исследование влияния мета-оптимизации на основе эволюционной адаптации на повышение эффективности

алгоритмов. Зависимость эффективности алгоритмов роевого интеллекта от используемых значений параметров была подтверждена вычислительными экспериментами. При этом показано, что при настройке алгоритма только под одну задачу найденные значения коэффициентов будут отличаться от задачи к задаче. Предложенная эволюционная адаптация алгоритмов роя частиц и колонии муравьев повысила эффективность относительно средних решений на 0,3–12 % (повышение в среднем равное для обоих методов). По большинству задач (85 %) были найдены наилучшие известные решения. На отдельных задачах результаты отличаются от наилучших, но отклонение не превышает 2,5 %, среднее отклонение менее 1 %. Экспериментально подтверждена устойчивость адаптации, поскольку повышение эффективности адаптивных алгоритмов роевого интеллекта происходит не только для отдельных задач, под которые выполнена адаптация, но и для других задача того же класса. Показано, что после выполнения адаптации настроенные роевые алгоритмы можно применять для решения задач не только в режиме проектного планирования, но в оперативном управлении. По задачам из библиотеки Operational Reserch Library время для получения решения, близкого к оптимальному (отклонение 0–2 %), составляет менее 0,5 секунды на процессоре Core i7 без распараллеливания расчетов.

# 5. Разработанное программное обеспечение

Исследование эффективности алгоритмов решения задач оптимизации невозможно без использования средств вычислительной техники. Раздел 5 описывает этап практической реализации разработанных алгоритмов до уровня конечных программных продуктов. Подраздел 5.1 посвящен обоснованию выбора инструментальных средств реализации. В подразделе 5.2 рассматривается применение алгоритмов роевого интеллекта в различных программных комплексах, в частности существующих системах для моделирования электроэнергетических систем. Подраздел 5.3 описывает реализованный и зарегистрированный программный комплекс для решения задач календарного планирования, а также версию для создания учебных расписаний. Подраздел 5.4 описывает приложение, созданное для визуализации работы стохастических алгоритмов.

## 5.1. Выбор инструментальных средств разработки

Для практической реализации разработанных моделей и алгоритмов важен выбор наиболее подходящих инструментальных средств разработки. К ним относятся в первую очередь язык программирования и инструментальная среда разработки.

**Выбор языка программирования.** Задачи как проектирования, так и оперативного управления как в области календарного планирования работы, так и в области управления электрическими сетями, имеют свою специфику. Ниже приведены требования к таким программам [Матренин, 2012].

1. Надежность и безопасность приложений. Особенно это требование важно для ПО в области автоматического управления техническими системами.

2. Простота интеграции ПО с различными уже существующими системами.

3. Продуманный понятный интерфейс.

4. Высокая скорость вычислений при решении оптимизационных задач.

Выбранный язык программирования также должен быть хорошо документированным и широко используемым. Всем вышеперечисленным требованиям лучше других удовлетворяют следующие языки программирования: C++, C#, Java.

Использование платформы языка C# обеспечивает высокий уровень безопасности ПО [Гросс, 2009]. Он имеет одновременно простой синтаксис и производительность близкую к C++ [Троелсен, 2004]. Хотя C# ориентирован на ОС Windows, стандарт .NET позволяет создавать технологии для исполнения приложений .NET на различных платформах, например, проект Mono для Unix-подобных ОС.

Необходимо учитывать потребности в больших по объему вычислениях для накопления и анализа статистических данных об эффективности и скорости работы различных алгоритмов. Поэтому целесообразно реализовать наиболее трудоемкие вычисления на языке, который обеспечил бы большую скорость работы, а именно C++. Непосредственно для исследований не требуется создавать интерфейс пользователя, средства работы с базами данных и прочее, так как эта часть необходима только для анализа и разработки алгоритмов и, очевидно, не нужна в версии приложения для конечного пользователя.

Язык C++ является одним из наиболее распространенных и многофункциональных языков программирования. Содержит все необходимые средства создания эффективных программ широкого круга применения (утилиты, драйвера, сложные программные комплексы, библиотеки алгоритмов). Поддерживаются различные стили и технологии программирования, включая традиционное директивное программирование, объектно-ориентированное программирование. Язык C++ обладает возможностью работы на низком уровне. На языке C++ разрабатывают программы для самых различных платформ и систем. Язык спроектирован так, чтобы дать программисту максимальный контроль над всеми аспектами структуры и порядка исполнения программы. Программы, написанные на C++, в большинстве случаев уступают в скорости только созданным на ассемблере при условии реализации ассемблерного кода специалистом очень

высокого уровня под конкретную архитектуру. Таким образом, язык C++ позволяет обеспечить максимально возможную или близкую к максимальной скорость вычислений.

В итоге были выбраны язык C++ для выполнения вычислений и создания несложных пользовательских интерфейсов, и язык C# для создания более сложных пользовательских интерфейсов, соединения с надсистемой (системами управления БД, средствами генерации отчетов и т.д.) в программах, где это необходимо.

**Выбор среды разработки.** Наиболее эффективными средствами создания программ на C++ являются MS Visual Studio, Borland C++ Builder и Qt SDK. Был выбран инструментарий Qt, который позволяет разрабатывать программное обеспечение уровнем от небольших программ до крупных научных проектов и коммерческих систем, которое сможет работать в различных ОС (Mac OS X, Windows, семейство Linux и других) [Бланшет, 2008; Шлее, 2012]. Преимуществами среды Qt является свободное распространение, кроссплатформенность самого инструментария и созданного при его использовании программного обеспечения, высокая скорость работы, наличие всех необходимых средств для оптимизации программ: библиотеки структур, данных и алгоритмов, возможности ассемблерных вставок, средства для распараллеливания работы. Инструментарий Qt широко используется как в зарубежных, так и в российских компаниях и проектах. Ниже описаны преимущества использования Qt как инструмента для профессионального программирования на языках высокого уровня и как среды для обучения студентов [Матренин, 2013].

1. Open-source для некоммерческого использования.
2. Кроссплатформенность.
3. Наличие необходимых для профессиональной разработки ПО библиотек.
4. Эффективная разработка пользовательских интерфейсов.

Для создания приложений с использованием языка C# выбрана Microsofr Visual Studio как наиболее мощное средство создания приложений на этом языке.

## 5.2. Применение разработанных алгоритмов роевого интеллекта в программных комплексах

Существует множество различных подходов к описанию, реализации и применению алгоритмов роевого интеллекта, что затрудняет их применение на практике, особенно для неспециалистов в методах оптимизации и программировании. Часто нелегко понять, как именно применить тот или иной алгоритм к существующей математической модели задачи, реализованной, например, на языке программирования высокого уровня или в таких системах, как «Simulink» от «MatLab» или «Rastr Win».

Как показано во второй главе, все алгоритмы роевого интеллекта имеют единую структуру и отличаются высокой гибкостью применения, так что можно создать единый интерфейс между алгоритмами и математической моделью задачи оптимизации, который позволил бы быстро встраивать алгоритмы роевого интеллекта в модели различных объектов для выполнения оптимизации. При этом интерфейс не должен быть привязан к конкретному языку или парадигме программирования, но должен быть легко переносимым. Поэтому ниже дается ряд примеров программной реализации такого интерфейса.

### 5.2.1. Программный интерфейс взаимодействия алгоритмов роевого интеллекта и задачи оптимизации

**Объектное проектирование методов оптимизации.** В рамках объектно-ориентированного программирования каждый алгоритм оптимизации можно рассматривать как класс. Полезно использовать единые интерфейсы таких классов, чтобы применение того или иного алгоритма выполнялось единообразно. В этом случае можно применить шаблон проектирования «Template method», который задает общую структуру поведения связанных классов. При этом детали поведения отдельного алгоритма задаются в реализации дочерних классов. Общими для дочерних классов будут методы управления алгоритмами, настройки,

инициализации, перемещения частиц и возвращения результатов работы. Пример интерфейса базового класса на языке программирования C++ приведен ниже.

```
class Swarm
{
public:
Swarm(I_Task* ptr_task, int iteration_number, unsigned int swarm_size);
virtual void set_parameters(const std::vector<double> parameter_vector) = 0;
virtual void reset() = 0;
virtual double run();
virtual double get_best(std::vector<double>* solution_vector) const;
virtual ~Swarm() {}
//…
};
```

**Взаимодействие методов с решаемыми задачами.** Для обеспечения единообразной работы алгоритмов оптимизации с различными задачами можно реализовать некоторый интерфейс, обозначенный здесь как *I_Task*, имеющий метод *calc_criterion*, который соответствует некоторой оптимизируемой функции $f(X)$ и принимает управляющие переменные $X$, а возвращает значение критерия $f(X)$. На языке программирования C++ реализация такого интерфейса с помощью абстрактного класса может быть следующей:

```
class I_Task
{
public:
   virtual double calc_criterion(const std::vector<double>& parameter_vector) const = 0;
   virtual unsigned int get_dimension() const = 0;
   virtual ~I_Task() {}
};
```

В языке Java реализация интерфейса немного короче:

```
public interface I_Task
{
   double calc_criterion (double [] parameters);
   int get_dimension();
};
```

Затем в классе, представляющем модель задачи оптимизации, нужно реализовать метод *getCriterion*, а сам класс должен наследоваться от класса *ICriterionCalc* (в C++) или реализовать интерфейс *ICriterionCalc* (в Java), например для задачи Химмельблау:

```
//C++
```

```
class Himmelblau : public I_Task
{
public:
   virtual double calc_criterion(const std::vector<double>& parameter_vector) const;
   virtual unsigned int get_dimension() const;
};

double Himmelblau::calc_criterion(const std::vector<double>& parameter_vector) const
{
   double x0 = parameter_vector[0], x1 = parameter_vector[1];
   return std::pow((x0*x0 + x1 – 11.), 2) + pow((x0+ x1*x1 – 7.), 2);
}

double unsigned int get_dimension()  const { return 2; }


//Java
public class Himmelblau implements I_Task
{
   double calc_criterion (double [] parameters)
   {
      double x0 = arrDouble[0], x1 = arrDouble[1];
      return Math.pow((x0*x0 + x1 – 11.), 2) + Math.pow((x0+ x1*x1 – 7.), 2);
   }
   int get_dimension () { return 2; }
}
```

Сам класс, представляющий модель решаемой задачи оптимизации, может быть только оболочкой для сложного объекта управления. Но каким бы сложным не была внутренняя логика расчета критерия, для алгоритма оптимизации она целиком будет сокрыта интерфейсом *I_Task*. При необходимости в интерфейсе могут быть дополнительные методы, помимо вычисления критерия, их можно добавить в *I_Task*. Если в приведенном примере нужно будет заменить критерий, это никак не повлияет на интерфейс *I_Task* и код алгоритмов оптимизации. Выше в коде класса *Swarm* в конструктор передается указатель на объект класса *I_Task*. В результате, благодаря полиморфизму, в классах, реализующих алгоритмы оптимизации, можно вызывать вычисление критерия через метод *calc_criterion* интерфейса *I_Task* независимо от класса, в котором будет производиться это вычисление, как показано на диаграмме классов на рисунке 5.1 (Particle Swarm Optimization и Ant Colony Optimization представляют примеры классов, реализующих конкретные алгоритмы оптимизации). Таким образом, код

алгоритмов оптимизации напрямую никак не связан с кодом класса, реализующего решаемую задачу оптимизации. Код интерфейсов приведен в приложении В.

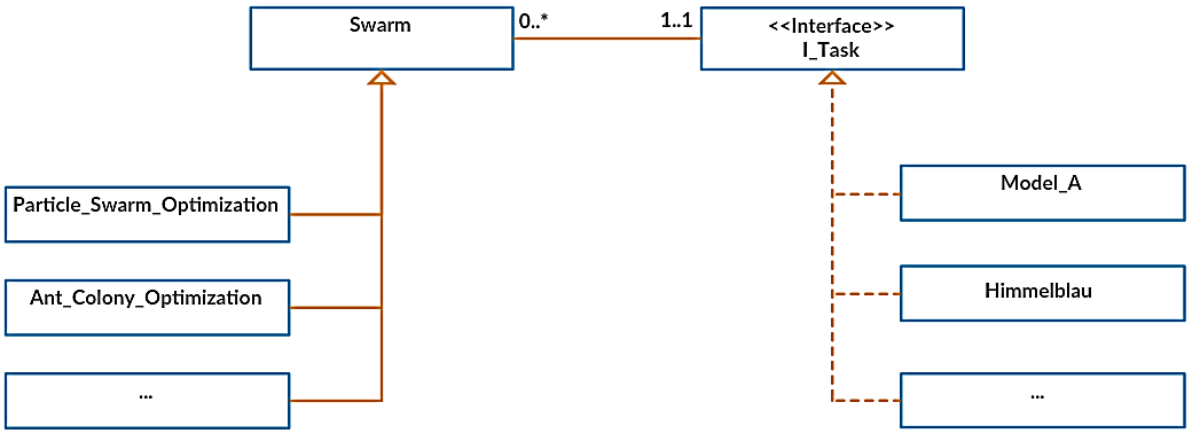



Рисунок 5.1 – Диаграмма классов, показывающая связь алгоритмов оптимизации и решаемых задач

### 5.2.2. Взаимодействие алгоритмов роевого интеллекта и «Simulink»

Систему моделирования «Simulink» широко часто используют в задачах из моделирования систем управления, химических процессов, систем связи, имитационного моделирования, энергосистем, силовой энергетики и во многих других областях, поскольку «Simulink» содержит множество библиотек для различных предметных областей, большой набор методов для проведения экспериментов с моделями, а процесс построения моделей является наглядным и простым. Для системы «Simulink» принцип организации взаимодействия между алгоритмом роевого интеллекта и моделью системы, предложенный в главе 2, остается неизменным. Необходимо добавить в модель всего три стандартных блока «Simulink».

1. Блок типа *simout*, с именем, например, *SimoutFX*, в который передавать полученное в результате моделирования значение критерия.

2. Константу *constant* с именем *Dimensionality*, в которой будет записана размерность (размер вектора $X$).

3. Блок типа *constant* для приема вектора значений критериев – *InputX*.

Таким образом, блок *Dimensionality* является аналогом метода *getDimensionality*(), а блоки *SimoutFX* и *InputX* обеспечат вычисление *f*(*X*), аналогично методу *getCriterion*(*double *arrayDouble*). Алгоритм роевого интеллекта реализуется в отдельном объекте (удобнее сделать это не в «Simulink», а текстовым представлением на языке программирования, используемом в «MatLab»). В этой реализации необходимо открыть модель из Simuink, получить размерность задачи, вызвав метод *get_param('model_name/Dimensionality', 'value')*. Затем на каждом шаге алгоритма для определения значения критерия *f*(*X*) необходимо вызвать следующие методы:

```
% задать вектор X
set_param('model_name/InputX', 'value', stringX);
%stringX – это строковое представление вектора X
% запустить расчет критерия и получить его значение
SimOut = sim('modle_name', 'SaveOutPut', 'on')
% прочитать результат
output = SimOut.find('SimoutFX').signals.values(1);
```

Модель для решения диофантова уравнения

$$20x_0 - 70x_1 + 40x_2 + 90x_3 - 35 = 0,$$

построенная в «Simulink» по описанному принципу, показана на рисунке 5.2.

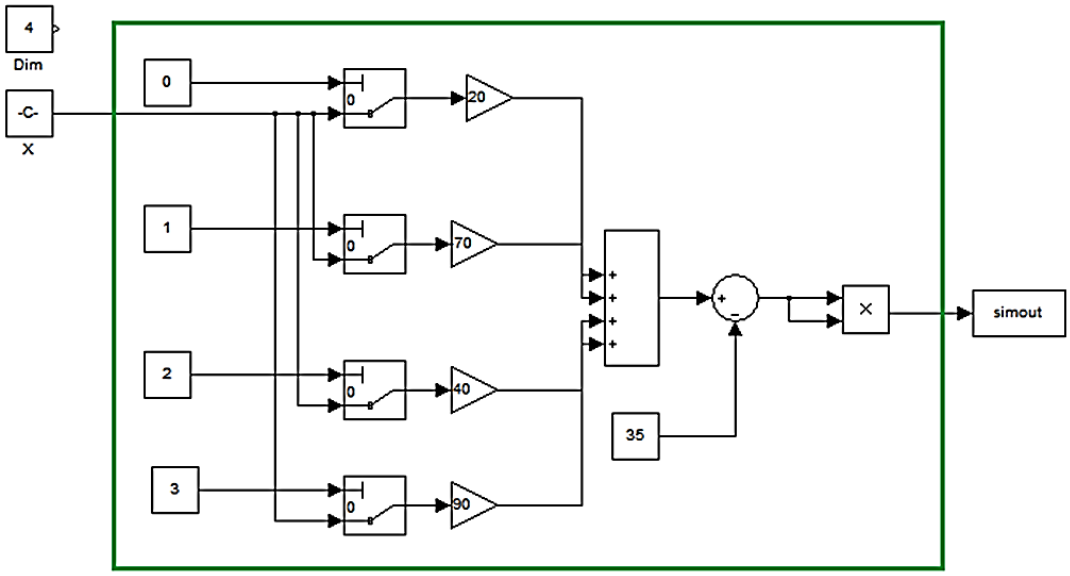



Рисунок 5.2– Реализация интерфейса в «Simulink» для применения роевых алгоритмов

### 5.2.3. Взаимодействие алгоритмов роевого интеллекта и «RastrWin3»

**Обзор программных средств расчета режима электросистем.** Оптимизация и управление систем электроснабжения требует расчетов режимов систем. При этом математические модели, позволяющие выполнять такие расчеты, слишком сложны для решения вручную. В настоящее время в России используется ряд программных комплексов для расчетов установившихся режимов электрических сетей [Непша, 2013]:

- «RastrWin», Екатеринбургский общественный фонд «Фонд им. Д.А. Арзамасцева»;
- «АНАРЭС-2000», ООО "ИДУЭС" и ЗАО «Энергетические технологии»;
- «SDO-6», Сибирский Энергетический институт им. Мелентьева СО РАН;
- «PSS/E», компания «Siemens»;
- «EUROSTAG», компания «Tractebel Engineering»;
- «Космос», ЗАО «Институт энергетических систем»;
- «DigSilent», компания «DIgSILENT GmbH».

Все указанные продукты позволяют представлять информацию о сетях как в табличном, так и в графическом виде, выполнять расчет напряжений в узлах сети, потокораспределения и потерь мощности в линиях.

При этом наиболее функциональным являются системы «Digisilent», «PSS/E», «RastrWin» и «АНАРЭС-2000» [Непша, 2013]. Однако, немецкие системы «Digisilent», «PSS/E» не адаптированы под российские регламенты и стандарты.

Интеграция разработанных алгоритмов оптимизации и программных систем возможна в трех случаях:

1) предоставление разработчиками системы ее исходных кодов;

2) наличие в системе специального интерфейса, позволяющего осуществлять обмен данными и командами между системой и сторонним приложением или плагинами;

3) реализация в системе механизма внедрения запрограммированных алгоритмов через макросы.

Первый путь дает наибольшие возможности, но при этом является самым трудоемким и невыгодным для компаний, создавших систему, так как открывает легкий путь для нелицензионного распространения копий и плагиата разработанных решений. Второй путь является наиболее простым для интеграции, но создание требуемого интерфейса сложно для реализации. Третий путь отчасти лишен указанных недостатков двух первых, а его минус заключается в необходимости реализации алгоритмов на том языке программирования, который задан системой.

Среди рассмотренных систем, адаптированных под российские сети, только в «RastrWin» реализована возможность интеграции. При этом в «RastrWin» применен третий путь, реализованный на базе макропрограммирования по технологии «Windows Scripting Host» (WSH). Кроме того, «RastrWin» используется более чем в 262 организациях и 26 образовательных учреждениях (на 31.10.2016 по данным сайта «Программный комплекс Rastr WIN») и имеет бесплатную студенческую лицензию. По совокупности критериев комплекс «RastrWin» оказался очевидно наилучшим вариантом системы для внедрения алгоритмов роевого интеллекта в задачи оптимизации систем электроснабжения.

**Макростудия «RastrWin».** Макросы в «RastrWin» используются для автоматизации выполнения последовательных действий и написания пользовательских программ. Макросы создаются на языке Visual Basic Script, который разработан компанией «Microsoft» и широко используется в офисных приложениях этой компании. Созданные макросы можно рассматривать как последовательности команд, выполняемых приложением. Среди них могут быть операции для выполнения расчетов непосредственно в макросе, так и обращения к ядру «RastrWin» для получения или установки параметров сети, выполнения расчетов режима и выполнения других функций.

**Получение и изменение значений параметров элементов схемы.** Для получения данных об узлах сети, таких как напряжения, активные и реактивные мощности нагрузок и генераторов, расчетные углы напряжения, необходимо в начале работы выполнить команды:

```
set nodes = Rastr.Tables("node")
set nodes_voltage = nodes.Cols("vras")
set nodes_gen_p = nodes.Cols("pg")
set nodes_gen_q = nodes.Cols("qg")
set nodes_delta = nodes.Cols("delta")
```

В процессе работы конкретное значение, например, напряжения в *i*-м узле можно определить или задать:

```
 v = nodes_voltage.z(i)
nodes_voltage.z(i) = v
```

Аналогично происходит работа с данными об узлах (Rastr.Tables("vetv")), трансформаторах (Rastr.Tables("ATtrans")) и других элементах сети.

**Работа алгоритмов оптимизации в «RastrWin».** Согласно разработанному интерфейсу между алгоритмами роевого интеллекта и решаемыми задачами, необходимо определить способ расчета критерия оптимальности $f(X)$, где $X$ – позиция некоторой частицы роевого алгоритма на очередной итерации работы. Для вычисления значения критерия в «RastrWin» необходимо выполнить расчет режима. Для этого в «RastrWin» предназначена команда rastr.rgm(""), которая возвращает не нулевое значение, если режим разошелся. Таким образом проверка ограничений по устойчивости сети выполняется автоматически. Учет ограничения в данной работы выполняется с помощью штрафной функции, так что решение задачи оптимизации, нарушающее устойчивость, получит значение критерия заведомо хуже допустимого решения. Таким образом, необходимо выполнить переход от позиции частицы $X$ к управляемыми (оптимизируемым) параметрам сети, например, номерам анцапф трансформаторов. Установить значения этих параметров, как показано в 5.2.3.3. Затем запустить расчет rastr.rgm(""), проверить сходимость режима и взять нужные выходные параметры, такие как потери активной мощности в линиях, отклонения напряжений в узлах. По ним рассчитать значение критерия (критериев) с учетом штрафных функций для ограничений. Полученное значение $f(X)$ передать в алгоритм роевого интеллекта. Эти шаги выполняются для каждой частицы на каждой итерации алгоритма. Важно отметить,

что после выполнения расчета режима и получения значений критерия или критериев оптимизации необходимо вернуть значения всех параметров сети к изначальным для проверки следующего решения. Это связано с тем, что «RastrWin» в процессе расчетов может вносить дополнительные изменения, например, в балансирующий узел. Поэтому необходимо перед расчетом режима сохранить напряжения, мощности генерации активной и реактивной мощности и расчетные углы напряжений во всех узлах. После выполнения расчет режима вернуть сохраненные значения.

Каждый раз после выполнения расчета режима «RastrWin» обновляет содержимое открытых таблиц с данными о сети и выводит ряд сообщений в консоль. Это значительно замедляет процесс оптимизации, так как на каждом шаге для каждой частицы больше процессорного времени будет уходить не на расчеты, а на обновления в графическом интерфейсе. Поэтому необходимо выполнить следующие команды перед началом оптимизации:

```
Rastr.LockEvent = True
Rastr.LogEnable = False
```

После завершения оптимизации необходимо выполнить эти же команды с обратными значениями для включения обновлений графического интерфейса и обновить содержимое показываемых таблиц вызовом команды

```
Rastr.SendChangeData 0,"","",0
```

## 5.3. Система календарного планирования

### 5.3.1. Приложение для решения задач календарного планирования

Разработанная и реализованная система предназначена для расчета календарных планов выполнения совокупности работ на определенных ресурсах, с использованием разработанных адаптивных алгоритмов роевого интеллекта. Версия системы с использованием адаптивного алгоритма колонии муравьев зарегистрирована в Реестре программ для ЭВМ Федеральной службы по интеллектуальной собственности под названием «Адаптивная система

календарного планирования методом колонии муравьев» (авторы Матренин П.В., Секаев В.Г., свидетельство о государственной регистрации № 2012614741. Версия программы, в которую включен, кроме адаптивного алгоритма колонии муравьев, адаптивный алгоритм роя частиц, зарегистрирована в Реестре программ для ЭВМ Федеральной службы по интеллектуальной собственности под названием «Система календарного планирования адаптивными алгоритмами роевого интеллекта» (№ 2013661938). Основные функции системы:

- ввод, редактирование, сохранение и загрузка данных о задачах КП с возможностью работы с разными формами хранения данных;

- решение задач КП в различных режимах;

- вывод на экран полученных результатов в виде графиков Ганта и таблиц;

- формирование выходных документов в виде единых отчетов, включающих исходные данные, полученные графики Ганта и данные о загрузке производственных мощностей и подробных планов по каждой работе, сохранение, просмотр и печать выходных документов;

- ввод, сохранение и редактирование параметров алгоритмов, используемых при расчетах;

- исследование влияния различных параметров алгоритма на получаемые результаты и поиск наилучших параметров с помощью снятия, сохранения и отображения статистических данных с использованием ГА;

На рисунке 5.3 приведена обобщенная UML-диаграмма вариантов использования системы планирования. Ниже приводится описания объектов, представленных на рисунке 5.3:

- Operator, человек, ответственный за составление планов, имеющий права вводить исходные данные, создавать, изменять планы и сохранять результаты своей работы в БД;

- ScheduleSystem, комплекс ПО, выполняющий необходимые расчеты и оценки для формирования планов;

- User, пользователь, имеющий доступ к планам, без права редактирования.

Варианты использования:

- InputData – ввод исходных данных для решения задачи КП;
- GenerateSchedule – генерация плана;
- ViewSchedule – просмотр плана;
- EditSchedule – ручное редактирования плана;
- SaveResult – сохранение результатов работы.

На рисунках 5.4–5.5 приведены примеры экранных форм приложения.

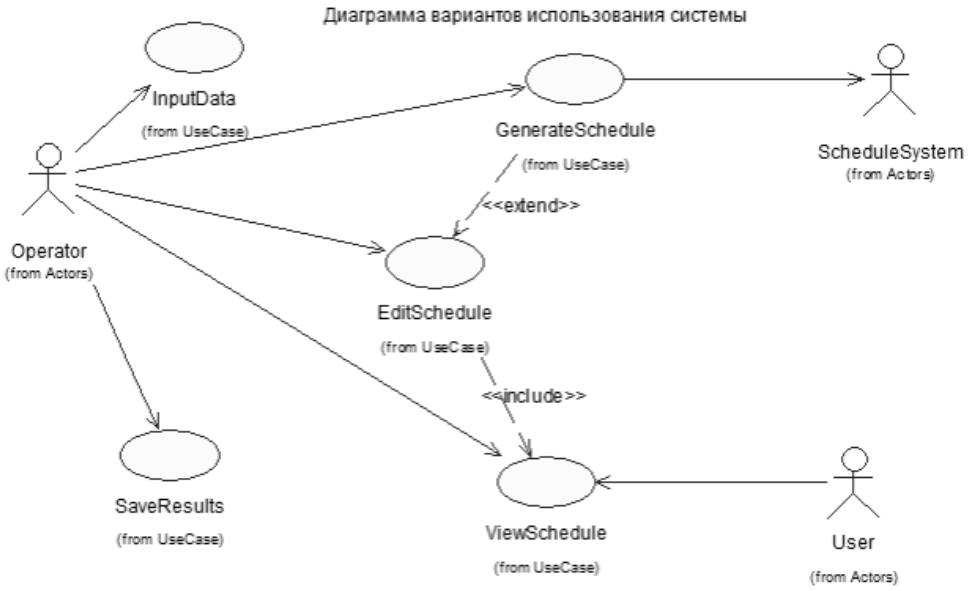



Рисунок 5.3 – Диаграмма вариантов использования

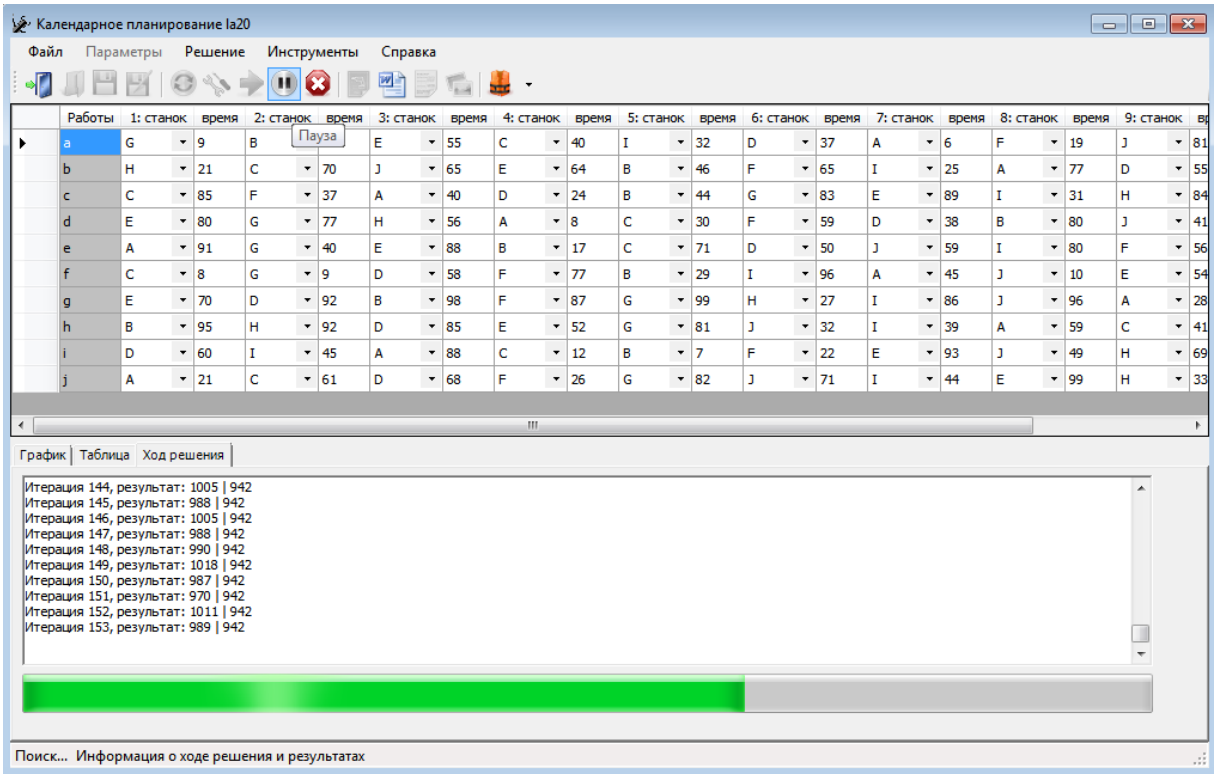



Рисунок 5.4 – Интерфейс приложения решения задач КП, поиск решения

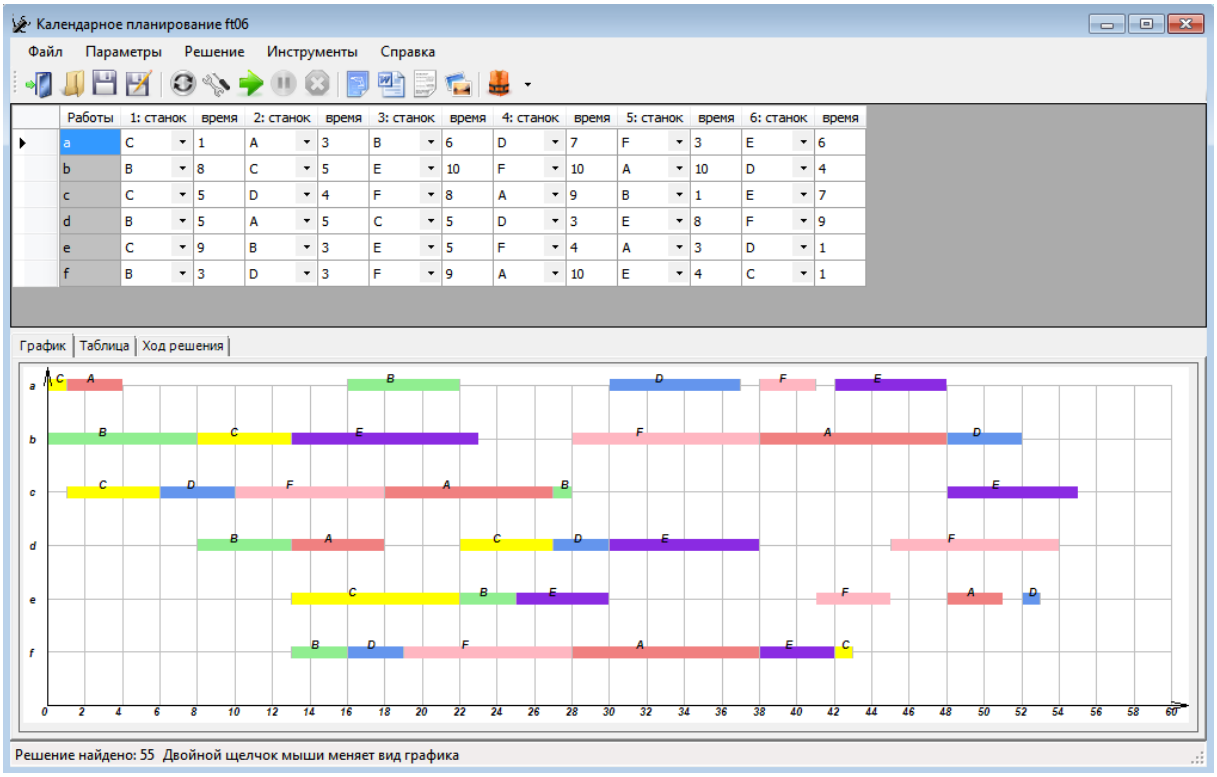



Рисунок 5.5 – Интерфейс приложения решения задач КП, график Ганта

### 5.3.2. Применение для составления учебных расписаний

Составление учебных расписаний – одна из наиболее актуальных и неформализованных частей теории расписаний. Применение разработанных алгоритмов для решения задач составления учебных расписаний, с одной стороны, является трудной задачей из-за необходимости создания математической модели, формализующей задачу, с другой стороны – получить необходимые входные данные работы в учебных заведениях часто проще, чем на предприятиях. Поэтому и в данной работе было проведено исследование задач составления учебных расписаний на базе Новосибирского архитектурно-строительного колледжа, в рамках исследования создано и апробировано программное обеспечение для составления учебных расписаний.

**Описание задачи составления учебного расписания.** Составление учебных расписаний может быть даже сложнее задач планирования на производстве в силу меньшей степени формализации ограничений и критериев [Береговых, 2009]. При этом размерность таких задач остается слишком большой

для применения детерминированных методов. Важно особенностью является необходимость получить в первую очередь не оптимальное, а хотя бы допустимое решение, так как из-за большого количества критериев точный оптимум находить не имеет смысла, ведь на оценку расписания влияют нормативные требования, пожелания преподавательского коллектива и учащихся. В результате возникает многокритериальная не полностью формализованная оптимизационная задача. Ее решение следует разделить на этап автоматического составления допустимого, приближенного к формальному критерию расписания, и этап ручной доработки, на котором выполняется коррекция с учетом дополнительных факторов, критериев и неформализованных особенностей.

**Алгоритм составления учебного расписания**. Для выполнения этапа автоматического составления расписания использована идея вычисления степени свободы каждого отдельного занятия, описанная в работе [Береговых, 2009]. Суть в определении ограничений, накладываемых на занятия. Среди ограничений могут быть требование особого оборудования, определенных аудиторий, число которых намного меньше обычных аудиторий, ограничения по времени проведения как в рамках дня, так по дням недели. Последнее требование больше связано с преподавателями и их графиками работы, особенно работающих по совместительству. Поэтому составление расписания следует начинать с занятий, имеющих наиболее жесткие ограничения, то есть минимальную степень свободы. Иначе с высокой вероятностью для них не будет найдено допустимое место в итоговом расписании. Занятия с наибольшей степенью свободы можно легко поместить в оставшиеся незаполненные позиции в конце составления расписания.

Данный подход позволяет учитывать сколь угодно сложные требования и получать приемлемые расписания достаточно быстро. Сложность заключается в выборе и настройке функции, по которой будет оцениваться свобода $i$-го занятия. Она должна учитывать: количество подходящих для проведения занятия аудиторий $a_i$, загруженность преподавателя как число его занятий в неделю $p_i$, загруженность группы учеников тоже как число занятий в неделю $g_i$. В этом случае оценка степени свободы $i$-го занятия может быть вычислена следующим образом:

$$S_i = \frac{a_i}{p_i * g_i}$$

Все занятия упорядочиваются по убыванию степени свободы, затем начиная с наименее свободного занятия, находить наиболее подходящие аудиторию и позицию во времени (день и номер занятия, возможно, номера учебных недель). После расстановки всех занятий определяется значение критерий как общего качества расписания. Приведенный алгоритм показан на рисунке 5.6 в виде UML-диаграммы деятельности. В работе для оптимизации расчетов использованы битовые маски, показывающие наличие свободных позиций в расписании для аудиторий, преподавателей, учебных групп [Матренин, 2013].

При работе данного алгоритма возникает задача оценки позиции для занятия исходя из всех требований, которые были формализованы. Поскольку требований много, оценка получается многокритериальной. Но решение должно быть принято в автоматическом режиме, поэтому в данной работе все требования сводятся в один критерий путем линейной свертки. Именно на этом этапе возникает задача, для решения которой были применены алгоритмы роевого интеллекта. Так как весовые коэффициенты, используемые в свертке критериев, могут принимать бесконечное множество значений, но от их выбора зависит получаемое расписание, то возникает задача их настройки. Полученная задача является оптимизационной задачей, в которой весовые коэффициенты выступают в роли управляемых переменных, а качество составленного расписания – в роли критерия.

**Описание разработанного программного обеспечения**. Схема работы с разработанным ПО не отличается от общей схемы, показанной на рисунке 5.3. Оператор составляет расписание в автоматизированном режиме с помощью системы планирования, может вносить в полученное расписание изменения вручную и сохранять его. Для ввода исходных данных созданы отдельные формы для редактирования списков групп, предметов, преподавателей и занятий (группа, предмет, преподаватель, количество часов в семестре).

Расписание можно составлять как в начале семестра, так и по ходу семестра на оставшееся время, поскольку часто необходимо вносить изменения в расписание из-за смены учителей и других непредвиденных обстоятельств.

Разработанное ПО является кроссплатформенным, написано на C++ и использованием фреймворка Qt. Для хранения данных используется СУБД SQLite, полученные расписания можно открывать с помощью различных редакторов таблиц, таких как Microsoft Excel или Open Calc. Экранные формы приложения приведены на рисунке 5.7.

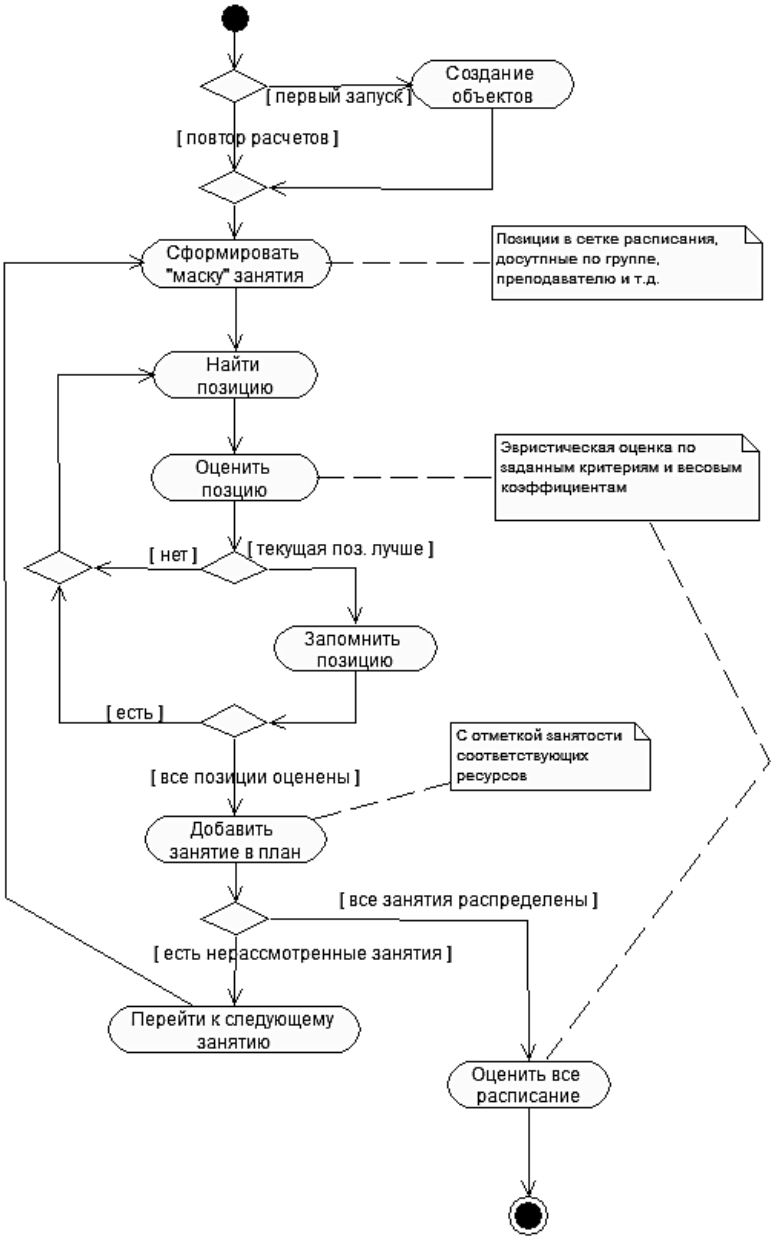
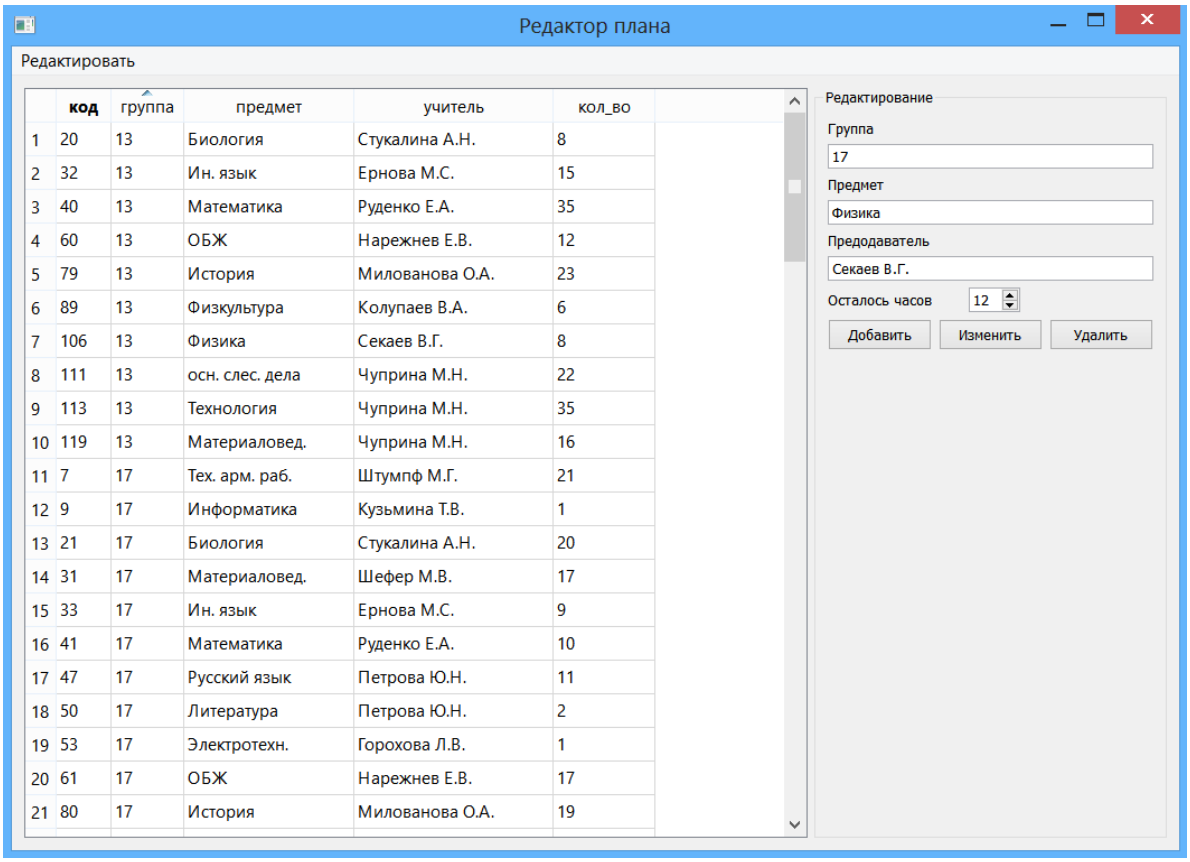



Рисунок 5.6 – UML диаграмма алгоритм составления учебного расписания




| | код | группа | предмет | учитель | кол_во |
|---|---|---|---|---|---|
| 1 | 20 | 13 | Биология | Стукалина А.Н. | 8 |
| 2 | 32 | 13 | Ин. язык | Ернова М.С. | 15 |
| 3 | 40 | 13 | Математика | Руденко Е.А. | 35 |
| 4 | 60 | 13 | ОБЖ | Нарежнев Е.В. | 12 |
| 5 | 79 | 13 | История | Милованова О.А. | 23 |
| 6 | 89 | 13 | Физкультура | Колупаев В.А. | 6 |
| 7 | 106 | 13 | Физика | Секаев В.Г. | 8 |
| 8 | 111 | 13 | осн. слес. дела | Чуприна М.Н. | 22 |
| 9 | 113 | 13 | Технология | Чуприна М.Н. | 35 |
| 10 | 119 | 13 | Материаловед. | Чуприна М.Н. | 16 |
| 11 | 7 | 17 | Тех. арм. раб. | Штумпф М.Г. | 21 |
| 12 | 9 | 17 | Информатика | Кузьмина Т.В. | 1 |
| 13 | 21 | 17 | Биология | Стукалина А.Н. | 20 |
| 14 | 31 | 17 | Материаловед. | Шефер М.В. | 17 |
| 15 | 33 | 17 | Ин. язык | Ернова М.С. | 9 |
| 16 | 41 | 17 | Математика | Руденко Е.А. | 10 |
| 17 | 47 | 17 | Русский язык | Петрова Ю.Н. | 11 |
| 18 | 50 | 17 | Литература | Петрова Ю.Н. | 2 |
| 19 | 53 | 17 | Электротехн. | Горохова Л.В. | 1 |
| 20 | 61 | 17 | ОБЖ | Нарежнев Е.В. | 17 |
| 21 | 80 | 17 | История | Милованова О.А. | 19 |

Рисунок 5.7 – Экранная форма редактора учебного плана

## 5.4. Визуализация популяционных алгоритмов

Особенностью рассматриваемых в работе роевых алгоритмов, как и других стохастических алгоритмов оптимизации, является их недетерминированное поведение, которое к тому же значительно зависит от их эвристических параметров. Эта особенность усложняет понимание работы алгоритмов, поскольку из математического описания не всегда можно ясно представить себе, как именно происходит процесс отыскания оптимального решения. Наиболее эффективным средством устранения этой сложности является создание приложения, которое бы наглядно показывало, как именно происходит работа различных алгоритмов, в чем их отличия, как процесс их работы зависит от топологии задачи оптимизации и от параметров алгоритмов.

**Требования к системе.** Исходя из указанного выше назначения приложения, были сформулированы перечисленные ниже требования:

1. Наглядная визуализация процесса работы стохастических алгоритмов.

2. Единообразие графических интерфейсов для каждого алгоритма, отличия в экранных формах для каждого из стохастических алгоритмов должны быть минимальными, но достаточными для учета их особенностей.

3. Возможность менять скорость визуализации шагов алгоритмов.

4. Возможность настраивать параметры алгоритмов как до начала, так и непосредственно в процессе работы, так как влияние значений параметров на эффективность очень велико, как показано в главах 3 и 4.

5. Высокое быстродействие, необходимое для корректной работы при задании большого количества частиц.

6. Возможность вводить свои функции для оптимизации, включающие в себя математические расчеты, ветвления, циклы и прочие основные конструкции языков программирования.

7. Масштабируемость графического интерфейса для возможности работы как на мониторах обычных ПК, так и на экранах с малой диагональю.

8. Кроссплатформенность для возможности работы в наиболее распространенных операционных системах, поскольку все чаще как отдельные студенты, так и образовательные учреждения уходят от использования ОС Windows в пользу UNIX систем.

**Описание созданного программного обеспечения.** В качестве средства создания приложения был выбран язык программирования C++ и инструментарий разработки Qt, согласно обоснованию, данному в 5.1. Выбор C++ и Qt позволили автоматически выполнить требования, относящиеся к интерфейсу, производительности и кроссплатформенности.

Для выполнения первого требования было решено ограничить рассматриваемые задачи следующей моделью:

$$\begin{cases} z = f(x, y) \\ x_{min} \le x \le x_{max} \\ y_{min} \le y \le y_{max} \\ z \to max \end{cases}$$

для таких задач процесс оптимизации можно визуализировать как процесс перемещения частиц в плоскости *XY*.

Каждый шаг работы выбранного алгоритма оптимизации запускается внутренним таймером приложения. Пользователь может менять интервалы отсчетов таймера, настраивая скорость визуализации, как указано в требовании 3. Использование таймера позволяет приложению одновременно выполнять расчеты, визуализацию и обрабатывать действия пользователя. Это позволяет менять параметры алгоритма, настройки самого приложения и даже решаемую задачу прямо во время работы алгоритмов оптимизации. Возможность пользователю вводить свои задачи приводит к необходимости проводить анализ введенных задач и выполнять расчеты по ним уже на этапе работы приложения. Была использована библиотека QtScript, позволяющая применять язык сценариев Qt Script. Введенные пользователями задачи проходят проверку и модификацию для повышения скорости расчетов. Так как задачу оптимизации нужно решать каждой из частиц на каждой итерации алгоритма, то исходная функция $z = f(x,y)$ модифицируется следующим образом:

$$Z_i = f(X_{i\,1}, X_{i\,2}),\ i = 1, \ldots, |S|$$

где, как было введено в подразделе 2.1, $X_i$ – позиция $i$-ой частицы, $X_{i\,1}$ и $X_{i\,2}$ – координаты частицы, $|S|$ - число частиц.

Благодаря этому интерпретировать задачу модулю QtScript придется только один раз, а не для каждой частицы, как было бы без данной модификации. Это позволяет в несколько раз увеличить число частиц алгоритма, при которых программа сохраняет достаточную для демонстрации быстроту работы.

Общее представление об интерфейсе приложений (рисунок 5.8) приведено для алгоритма роя частиц. Сверху расположена основная расчетная формула, ссылка на файл справки и метка для вывода результата. Ниже находится поле для ввода задачи, справа от которого расположен элемент для выбора функций списка. Большую часть формы занимает поле для визуализации процесса работы алгоритма, слева расположены элементы для ввода параметров. Текущее лучшее решение обозначается на поле красным крестиком.

На данный момент реализованы визуализации следующих стохастических алгоритмов: роя частиц, роя пчел, роя светлячков, имитации отжига и генетический. Два последних реализованы в приложении, чтобы наглядно продемонстрировать отличия в работе роевых алгоритмов от других стохастических алгоритмов.

Разработанные приложения используются на кафедрах «Автоматизированные системы управления» и «Системы электроснабжения предприятий» НГТУ. Приложения могут использоваться студентами самостоятельно по учебному пособию П.В. Матренина, М.Г. Грифа, В.Г, Секаева «Методы стохастической оптимизации» [Матренин, 2016а]. Приложения зарегистрированы в Реестре программ для ЭВМ Федеральной службы по интеллектуальной собственности (Свидетельство №2015613847).

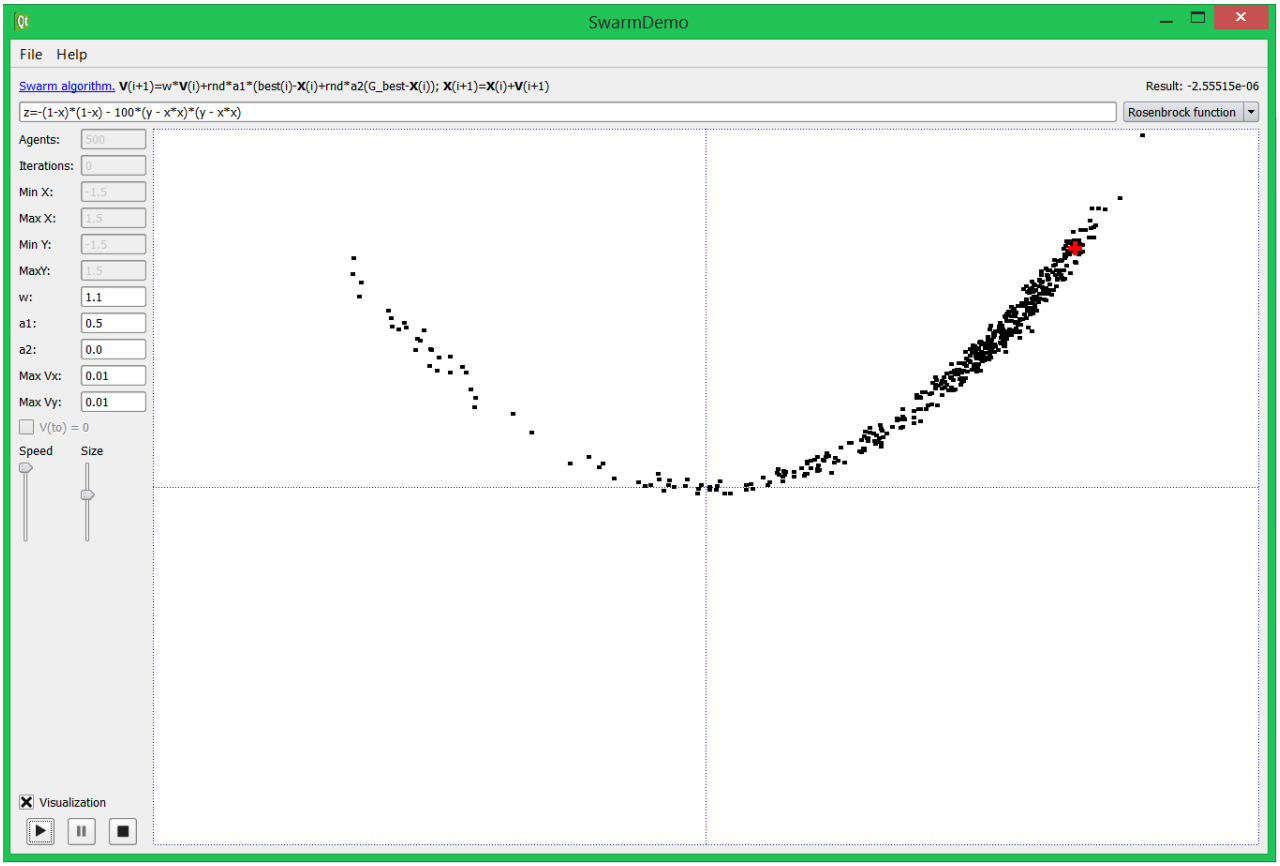



Рисунок 5.8 – Интерфейс приложения визуализации алгоритмов

## 5.5. Выводы по разделу 5

Был проведен анализ инструментальных средств и требований к программному обеспечению в проектах, применяющих алгоритмы роевого интеллекта для электротехнических систем, систем проектирования и управления. На основании результатов анализа были выбраны следующие средства и технологии: Qt и язык C++ для реализации вычислений и несложных приложений, Microsoft Visual Studio и язык C# для написания модулей крупных программных комплексов, таких как системы календарного планирования.

Благодаря разработанной схеме описания алгоритмов роевого интеллекта и их взаимодействия с задачами оптимизации появилась возможность легко применять различные алгоритмы роевого интеллекта для решения практических задач. Разработаны принципы программной реализации систем, использующих алгоритмы роевого интеллекта и приведены примеры реализации интерфейса между алгоритмом и задачей для языков программирования C++, Java, а также для системы моделирования «Simulink». Проведен анализ существующих программных комплексов для расчетов режимов электросетей. Была выбрана система «RastrWin3», позволяющая применять к сетям пользовательские алгоритмы оптимизации через механизм написания макросов на языке Visual Basic Script. Разработанный интерфейс между алгоритмами роевого интеллекта и задачами оптимизации был реализован на этом языке для применения алгоритмов в задачах оптимизации электросетей.

Была спроектирована, реализована и зарегистрирована система для решения задач календарного планирования, позволяющая редактировать файлы с исходными данными, составлять планы в различных режимах и выводить результаты в виде таблиц, графиков и отчетов. Достоинством данного программного приложения являются возможность адаптации к различным реальным задачам, благодаря универсальности и гибкости разработанных алгоритмов, низкие требования к системе, модульность и простота в

использовании. Была создана отдельная версия программного обеспечения для решения задач составления учебных расписаний.

Созданное по принципам системного описания алгоритмов роевого интеллекта приложение для визуализации применяется в рамках курсов, посвященных методам оптимизации и искусственному интеллекту на кафедрах АСУ и СЭСП НГТУ. На базе этого приложения и созданным системным описаниям алгоритмов роевого интеллекта написано и издано в НГТУ учебное пособие.

Разработанные комплексы программ зарегистрированы в Государственном Реестре программ для ЭВМ (Свидетельства о регистрации №201261474, №2013661938, №2015613847, №2018611639 в Приложении Б).

# ЗАКЛЮЧЕНИЕ

В настоящей диссертации разработаны математическая модель, обобщающая алгоритмы роевого интеллекта, и унифицированная схема применения роевых алгоритмов для оптимизации технических систем. Проведено исследование адаптационных свойств роевых алгоритмов и модифицированы известные методы мета-оптимизации. На основании выполненных исследований можно сделать следующие выводы и обобщения.

1. Проведен анализ оптимизационных задач проектирования и управления техническими системами, а также методов их решения. Выявлены следующие особенности рассматриваемых задач оптимизации: сложная топология пространства поиска решений, многокритериальность, многофакторность, наличие как дискретных, так и непрерывных управляемых переменных и ограничений, динамические изменения условий задач, высокая вычислительная сложность. Показано, что необходимо применять методы оптимизации, имеющие свойства гибкости, самоорганизации, позволяющие за приемлемое время получать решения, близкие к оптимальным. К ним относятся алгоритмы роевого интеллекта, эффективность которых основана на способах решения задач, созданных природой за миллиарды лет и при этом имеющих математические доказательства эффективности, основанные на теории конечных цепей Маркова.

2. Раскрыты причины трудностей в применении алгоритмов роевого интеллекта для решения оптимизационных задач на практике: отсутствие единой классификации и терминологии, отсутствие единого подхода к описанию и применению алгоритмов, необходимость настройки алгоритмов и методологические сложности в соединении алгоритмов и решаемых задач. Указанные трудности, обобщенно названные трудностями адаптации, снижают эффективность применения алгоритмов роевого интеллекта, что в свою очередь, не позволяет в полной мере улучшать показатели технических систем при решении оптимизационных задач.

3. Разработана новая модель описания, реализации и применения алгоритмов роевого интеллекта на основе проведенного системного анализа роевых алгоритмов. Введены правила классификации и единая терминология, упрощающие анализ и повышающие эффективность создания и применения роевых алгоритмов. Дано системное определение алгоритмам роевого интеллекта, описаны общие для них принципы и схема работы, что позволило выделить общие и уникальные для каждого алгоритма приемы, обеспечивающие их большую или меньшую эффективность в различных классах оптимизационных задач. Предложенная модель алгоритмов роевого интеллекта обеспечивает стандартизацию, повышение гибкости и переносимости создаваемого программного обеспечения, повышение скорости разработки.

4. Проведена модернизация существующих методов мета-оптимизации алгоритмов роевого интеллекта. Показано, что эффективная адаптация алгоритмов роевого интеллекта для решения практических задач имеет два основных аспекта. Первым является устранение тесных зависимостей между реализацией алгоритмов и моделями оптимизируемых систем. Для этого введен интерфейс между алгоритмами и задачами оптимизации, позволяющий выполнить их соединение без необходимости вносить изменения в математические модели алгоритмов и в модели оптимизируемых систем. Необходимые особенности, такие как масштабирование значений управляемых переменных, учет ограничений и дискретность, учитываются в интерфейсе. Задача оптимизации становится для алгоритма черным ящиком, как и алгоритм для задачи. Вторым аспектом адаптации является настройка значений эвристических коэффициентов роевых алгоритмов под определенный класс задач. Показано, что эффективным методом такой настройки является мета-оптимизация на базе генетического алгоритма по аналогии с эволюционной адаптацией в природе. Проведенные вычислительные эксперименты подтвердили, что предложенный метод адаптации обеспечивает повышение эффективности не только для задач, для которых выполнена адаптация, но и для других задач данного подкласса.

5. Раскрыты возможности применения адаптивных алгоритмов роевого интеллекта для решения задач оптимизации в проектировании и управлении электроэнергетическими системами. Решена многокритериальная гибридная задача определения положений и мощностей компенсирующих установок для компенсации реактивной мощности. Задача имеет свойства как дискретной, так и непрерывной оптимизации, поскольку мощности установок непрерывны, но места их допустимого размещения дискретны. Данная задача рассмотрена как одно- и двухкритериальная, как проектная и как задача оперативного управления. Показано, что при решении задачи как многокритериальной, лицо, принимающее решения, получает больше информации о вариантах улучшения системы, что особенно важно, если среди критериев часть влияет на краткосрочные показатели, часть на долгосрочные и необходимо соблюдать баланс тактических и стратегических целей. Эксперименты проводились для различных электроэнергетических систем: подсистема электроснабжения Ангарского электролизного химического комбината, подсистема электроснабжения ОА «Уральский электрохимический комбинат», фрагмента энергетической системы Таджикистана. Применение адаптивных алгоритмов роевого интеллекта позволяет снизить потери активной мощности на 5–20 %. Проведено сравнение адаптивных алгоритмов роевого интеллекта с роевыми алгоритмами без разработанной адаптации и другими оптимизационными методами (генетический алгоритм, градиентные методы). Эффективность решений адаптивных алгоритмов выше на 2–9 %, чем роевых алгоритмов без адаптации. Показано, что эффективность решений адаптивных алгоритмов выше на 2–9 %, чем роевых алгоритмов без адаптации. При этом существующие программные аналоги, применяемые для оптимизации электроэнергетических систем, не позволяют решать многокритериальные задачи или задавать все необходимые ограничения. Кроме того, были на практике подтверждены выводы теоретического анализа о преимуществе алгоритма роя пчел в задачах динамической оптимизации, поскольку частицы роя пчел при правильно настроенных параметрах алгоритма всегда рассредоточены по нескольким альтернативным окрестностям в

пространстве поиска решений. Адаптивные алгоритмы роевого интеллекта использованы в решении задачи выбора коэффициентов трансформации в системах электроснабжения 110–220 кВ. Проведенные вычислительные эксперименты показали, что при оптимизации с использованием адаптивных алгоритмов роевого интеллекта в рассмотренной системе можно добиться снижения потерь активной мощности с 48,01 МВт до 45,83 МВт (4,5 %). Выявлено, что рассматриваемый класс задач обладает сложной топологией, при которой оказываются малоэффективными применяемые в данной задаче методы, основанные на направленном переборе, эвристических правилах или нечеткой логике. Показано, что эффект от применения адаптивного алгоритма роя частиц с настраиваемым ограничением скорости на 28,9 % выше, чем эффект от указанных методов для рассмотренной электроэнергетической системы.

6. Работоспособность и эффективность разработанных адаптивных алгоритмов роевого интеллекта протестирована на задачах оптимизации из Operational Research Library. Разработана схема для применения роевых алгоритмов, выполняющих поиск как в непрерывном пространстве поиска решений, так и в дискретном. Установлено, что алгоритмы роевого интеллекта могут быть успешно применены для решения задач календарного планирования. Вычислительными экспериментами показано, что предложенная эволюционная адаптация повышает эффективность роевых алгоритмов на величину до 12 %. По 85 % задач получены результаты, совпадающие с наилучшими результатами в опубликованных работах других авторов, по отдельным задачам отклонение составило не более 2,5 %. Подтверждена устойчивость адаптации, поскольку повышение эффективности адаптивных алгоритмов роевого интеллекта происходит не только для отдельных задач, под которые выполнена адаптация, но и для других задач того же класса.

7. Разработан унифицированный интерфейс между программными реализациями алгоритмов роевого интеллекта и моделями оптимизируемых объектов для различных языков программирования и программных комплексов, используемых для моделирования электроэнергетических систем. Разработанные

программные библиотеки позволяют прикладным специалистам легко применять алгоритмы роевого интеллекта в решении оптимизационных задач. Для этого не требуется внесение изменений в модели оптимизируемых систем, созданных в специализированных широко используемых средах моделирования, таких как «Matlab», «Simulink», «RastrWin», а также в программных комплексах, написанных на языках программирования высокого уровня. Разработаны и внедрены программные приложения для демонстрации применения разработанных алгоритмов. Создан программный комплекс для решения задач календарного планирования, позволяющий работать с исходными данными задач календарного планирования, решать их различными методами и выводить результаты решений в различных форматах. Для анализа работы стохастических алгоритмов оптимизации разработано и внедрено в учебный процесс интерактивное приложение, выполняющее визуализацию их работы. Приложение используется в рамках учебных дисциплин, посвященных методам оптимизации и системам искусственного интеллекта.

Перспективным направлением дальнейших исследований является совершенствование разработанных методов и алгоритмов (повышение точности получаемых решений и увеличение быстродействия) для применения в оптимизации сложных многокритериальных технических систем с учетом их стохастических и динамических свойств, а также в задачах управления в реальном времени. Использование разрабатываемых адаптивных алгоритмов роевого интеллекта в указанных задачах является перспективным средством повышения качества функционирования технических систем и их экономической эффективности. На данный момент практическими направлениями развития работы являются решение задач оптимизации в управлении потоками электроэнергии в системах распределенной генерации и в структурно-параметрическом проектировании трансформаторов с высокотемпературной сверхпроводящей обмоткой.

# СПИСОК СОКРАЩЕНИЙ

АКМ – адаптивный алгоритм колонии муравьев

АРП – адаптивный алгоритм роя пчел

АРЧ – адаптивный алгоритм роя частиц

ДПФ – дискретное преобразование Фурье

ГА – генетический алгоритм

РИ – роевой интеллект

КП – календарное планирование

КУ – компенсирующая установка (источник реактивной мощности)

ОС – операционная система

ПО – программное обеспечение

РЧ – рой частиц

РП – рой пчел

# СПИСОК ЛИТЕРАТУРЫ

1. Акулич И.Л. Математическое программирование в примерах и задачах: учеб. пособие для студентов эконом. спец. вузов. – М.: Высш. шк., 1986. 319 с.

2. Анфилофьев А.А., Ходашинский И.А., Бардамова М.Б., Сарин К.С. Метаэвристические методы отбора информативных классифицирующих признаков // Информационные и математические технологии в науке и управлении. 2017. № 2(6). С. 11–20.

3. Баранюк В.В., Смирнова О.С. Роевой интеллект как одна из частей онтологической модели бионических технологий // International Journal of Open Information Technologies. 2015. Vol. 3, is. 12. P. 13–16.

4. Батищев Д.И., Неймарк Е.А., Старостин Н.В. Применение генетических алгоритмов к решению задач дискретной оптимизации [Электронный ресурс]. 2007. 85 с. URL: http://www.unn.ru/pages/e-library/aids/2007/15.pdf (дата обращения: 15.09.2014).

5. Белл Э.Т. Творцы математики. Предшественники современной математики. – М.: Книга по требованию, 2012. 253 с.

6. Береговых Ю.В., Васильев Б.А., Володин Н.А. Алгоритм составления расписания занятий // Искусственный интеллект. 2009. № 2. С. 50–57.

7. Береснев В.Л., Гончаров Е.Н. Приближенный алгоритм для задачи минимизации полиномов от булевых переменных // Дискретный анализ и исследование операции. 1998. Серия 2. Т. 5, № 2. С. 3–19.

8. Бланшет Ж., Саммерфилд М. Qt4: программирование GUI на C++: пер. с англ. – М.: КУДИЦ-ПРЕСС, 2008. 736 с.

9. Бураков М.В. Нейронные сети и нейроконтроллеры: учеб. пособие. – СПб.: ГУАП, 2013. 284 с.

10. Ватутин Э.И., Титов В.С., Емельянов С.Г. Основы дискретной комбинаторной оптимизации: учеб. пособие. – М: Аргамак-Медиа, 2016. 270 с.

11. Вентцель Е.С. Исследование операций: задачи, принципы, методология. 2-е изд. – М.: Наука, 1988. 208 с.

12. Винер Н. Кибернетика, или Управление и связь в животном и машине: пер. с англ. И.В. Соловьева и Г.Н. Поварова / под ред. Г.Н. Поварова. 2-е изд. – М.: Наука; Главная редакция изданий для зарубежных стран, 1983. 344 с.

13. Гараничин О.Н. Введение в методы стохастической оптимизации и оценивания: учеб. пособие. – СПб.: Изд-во С.-Петербургского ун-та, 2003. 131 с.

14. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы. 2-е изд., испр. и доп. – М.: ФИЗМАТЛИТ, 2006. 320 с.

15. Гладков Л.А., Курейчик В.В., Курейчик В.М., Сороколетов П.В. Биоинспирированные методы в оптимизации: монография. – М: Физматлит, 2009. 384 с.

16. Глушков В.М., Грибин В.П. Компенсация реактивной мощности в электроустановках промышленных предприятий. – М.: Энергия, 1975. 104 с.

17. Гончаров Е.Н., Ерзин А.И., Залюбовский В.В. Исследование операций. Примеры и задачи. – Новосибирск: Изд-во Новосибирского государственного технического ун-та, 2005. 78 с.

18. Гросс К. C# 2008: пер. с англ. – СПб.: БВХ-Петербург, 2009. 576 с.

19. Емельянов В.В., Курейчик В.В., Курейчик В.М. Теория и практика эволюционного моделирования. – М: Физматлит, 2003. 432 с.

20. Железко Ю.С. Компенсация реактивной мощности в сложных электрических системах. – М.: Энергоиздат, 1981. 200 с.

21. Жилинскас А., Шалтянис В. Поиск оптимума: компьютер расширяет возможности. – М.: Наука, 1989. 128 с.

22. Жмак Е.И., Манусов В.З. Обоснование принципа нечеткого регулирования напряжения с помощью РПН трансформаторов // Электроэнергетика: Сб. науч. тр. – Новосибирск: Изд-во Новосибирского государственного технического ун-та, 2002. С. 32–42.

23. Зайцев А.А, Курейчик В.В., Полупанов А.А. Обзор эволюционных методов оптимизации на основе роевого интеллекта // Известия Южного федерального ун-та. 2010. № 12. С. 7–12.

24. Зельдович Я.Б., Мышкис А.Д. Элементы прикладной математики. 3-е изд., перераб и доп. – М.: Наука, Глав. ред. физ-мат. лит., 1972. 592 с.

25. Кабышев А.В. Компенсация реактивной мощности в электроустановках промышленных предприятий: учебное пособие. – Томск: Изд-во Томского политехнического ун-та, 2012. 234 с.

26. Карпенко А.П. Популяционные алгоритмы глобальной оптимизации. Обзор новых и малоизвестных алгоритмов // Приложение к журналу «Информационные технологии». 2012. № 7. С. 1–32.

27. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ: пер. с англ. 2-е изд. – М.: Издательский дом «Вильямс», 2005. 1296 с.

28. Костин Н.С., Грицай А.С. Выбор оптимального доверительного интервала в задачах краткосрочного прогнозирования электропотребления // Омский научный вестник. Сер. Приборы, машины и технологии. 2016. № 3 (147). С. 63–67.

29. Кочегурова Е.А. Мартынов Я.А., Мартынова Ю.А., Цапко С.Г. Алгоритм муравьиных колоний для задачи проектирования рациональных маршрутных сетей городского пассажирского транспорта // Вестник СибГУТИ. 2014. № 3. С. 89–100.

30. Кочетов Ю.А. Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения: Сборник лекций молодежных и научных школ по дискретной математике и ее приложениям. – М.: Издательство центра прикладных исследований при механико-математическом факультете МГУ, 2000. С. 87–117.

31. Кочетов Ю.А., Столяр А.А. Новые жадные эвристики для задачи календарного планирования с ограниченными ресурсами // Дискретный анализ и исследование операций. 2005. Т. 12, № 1. С. 12–36.

32. Кочетов Ю.А., Панин А.А., Плясунов А.В. Генетический локальный поиск и сложность аппроксимации задачи балансировки нагрузки на серверы // Автоматика и телемеханика. 2017. № 3. С. 51–62.

33. Красник В.В. Управление электрохозяйством предприятий. Производственно-практическое пособие. – М.: Изд-во НЦ ЭНАС, 2004. 152 с.

34. Красов А.В. Теория информационных процессов и систем [Электронный ресурс]. 2014. 117 с. URL: http://www2.mts-sut.ru/kafedr/ibts/doc/tips.pdf (дата обращения: 10.07.2014).

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

36. Курейчик В.М., Кажаров А.А. Использование роевого интеллекта в решении NP-трудных задач // Известия ЮФУ. Технические науки. 2011. № 7 (120). С. 30–37.

37. Лазарев А.А., Гафалов Е.Р. Теория расписаний. Задачи и алгоритмы. – М.: Изд-во МГУ им. М.В. Ломоносова, 2011. 222 с.

38. Лебедев Б.К., Шашелов А.А. Исследование механизмом муравьиной адаптации при решении задачи покрытия функциональной схемы // Труды Конгресса по интеллектуальным системам и информационным технологиям «IS_IT'10». Пос. Дивноморское, Краснодраский край, 02–10 сентября 2010 г. М.: Физлитмат, 2010. Т. 3. С. 118–127.

39. Лебедев В.Б. Метод пчелиной колонии в комбинаторных задачах на графах // XII национальная конференция по искусственному интеллекту с международным участием (КИИ-212). Белгород, 16–20 октября 2012 г. Белгород: Изд-во Белгородского государственного технического ун-та, 2012. Т. 2. С. 414–421.

40. Лебедев В.Б., Лебедев О.Б. Роевой интеллект на основе интеграции моделей адаптивного поведения муравьиной и пчелиной колоний // Известия ЮФУ. Технические науки. 2013. № 7 (144). С. 41–47.

41. Леванов Т.В., Усько О.В. Алгоритм муравьиной колонии для одной задачи размещения с ограничениями на мощность производства // Вестник УГАТУ. 2010. Т. 14, № 2(37). С.202–208.

42. Лопатин А. Метод отжига [Электронный ресурс]. 2013. 10 с. URL: http://wtsim.googlecode.com/hg-history/default/docs/52.pdf (дата обращения: 22.10.2014).

43. Любченко В.Я., Павлюченко Д.А. Оптимизация коэффициентов трансформации подстанций на основе генетического алгоритма // Транспорт: наука, техника, управление. 2008. № 6. С. 30–33.

44. Манусов В.З., Матренин П.В., Орлов Д.В. (2017a) Оптимизация коэффициентов трансформации с применением алгоритмов направленного перебора и роевого интеллекта // Problems of regional energy. 2017. № 1 (33). С. 15–23.

45. Манусов В.З., Третьякова Е.С. (2017b) Глубокая компенсация реактивной мощности в системах электроснабжения производства // Энергобезопасность и энергосбережение. 2017. № 4. С. 33–38.

46. Матренин П.В., Секаев В.Г. (2013a) Системное описание алгоритмов роевого интеллекта // Программная инженерия. 2013. № 12. С. 39–45.

47. Матренин П.В. (2013b) Проектирование системы составления учебных расписаний с использованием метаэвристических алгоритмов // Новый университет. Серия «Технические науки». 2013. № 7(17). С. 63–70.

48. Матренин П.В., Секаев В.Г. (2013c) Оптимизация адаптивного алгоритма муравьиной колонии на примере задачи календарного планирования // Программная инженерия. 2013. № 4. С. 34–40.

49. Матренин П.В. Среда Qt как основной инструментарий разработки программного обеспечения в рамках обучения программированию // Информатика и образование. 2014. № 2 (251). С. 42–45.

50. Матренин П.В. Описание и реализация алгоритмов роевого интеллекта с использованием системного подхода // Программная инженерия. 2015. № 3. С. 27–34.

51. Матренин П.В., Гриф М.Г., Секаев В.Г. Методы стохастической оптимизации: учебное пособие. – Новосибирск: Изд-во Новосибирского государственного технического ун-та, 2016. 67 с.

52. Непша Ф.С., Отдельнова Г.В., Савинкина О.А. Сравнение функциональных возможностей существующих программных средств расчета и

анализа электрических режимов // Вестник Кузбасского государственного технического ун-та. 2013. № 2. С. 116–118.

53. Никифорова А.В., Канев В.С., Полетайкин А.Н. Методика оптимального планирования продвижения услуг связи на региональный рынок при помощи пчелиного алгоритма // Труды 12-й международной азиатской школы-семинара «Проблемы оптимизации сложных систем». Новосибирск, 12–16 декабря 2016 г. Новосибирск: Изд-во Новосибирского государственного ун-та, 2016. С. 442–446.

54. Норенков И.П. Эвристики и их комбинации в генетических методах дискретной оптимизации // Информационные технологии.1999. № 1. С. 2–7.

55. Оптимизация в электроэнергетических системах. Практические занятия: учеб. пособие для вузов / под ред. А. Г. Русиной. – М.: Издательство Юрайт, 2017. 158 с.

56. Панченко Т.В. Генетические алгоритмы: учебно-методическое пособие / под ред. Ю.Ю. Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. 87с.

57. Петров Ю.П. История и философия науки. Математика, вычислительная техника, информатика. – СПб.: БХВ-Петербург, 2005. 448 с.

58. Плотников А.А., Блок И.Н. Применение генетических алгоритмов комбинирования эвристик и метода роя частиц в задачах построения оптимального расписания // Автоматизированные системы и информационные технологии: Сборник трудов Российской научно-практической конференции. Новосибирск, 22 сентября 2011 г. Новосибирск: Изд-во Новосибирского государственного технического ун-та, 2011. С. 180–189.

59. Потапов В.И., Грицай А.С., Тюньков Д.А. Спектральный анализ ретроспективных данных ООО «Омская энергосбытовая компания» об электропотреблении // Омский научный вестник. 2016. № 5 (149). С. 74–76.

60. Растригин Л.А. Статистические методы поиска. – М: Наука, 1968. 376 с.

61. Родионов А.С., Сакерин А.В., Мигов Д.А. Некоторые вопросы параллельной реализации полного перебора // Вестник СибГУТИ. 2014. № 4. С. 80–85.

62. Руденко Ю.Н. Автоматизация диспетчерского управления в электроэнергетике / под общ. ред. Ю.Н. Руденко и В.А. Семенова. – М.: Издательство МЭИ, 2000. 648 с.

63. Рыжов А.П., В.С. Иванов Моделирование энергетических систем в рамках многоагентного подхода // Интеллектуальные системы. Теория и приложения. 2014. Т. 18, № 4. С. 39–46.

64. Самигулина Г.А., Масимканова Ж.А. Обзор современных методов роевого интеллекта для компьютерного молекулярного дизайна лекарственных препаратов // Проблемы информатики. 2016. № 2(31). С. 50–61.

65. Сарин К.С., Ходашинский И.А. Метод Чиу для отбора информативных признаков нечетких классификаторов // Информатика и системы управления. 2017. № 3(53). С. 84–95.

66. Секаев В.Г. Опыт использования генетических алгоритмов при построении оптимальных расписаний обслуживающих систем // Сборник научных трудов НГТУ. 2005. № 2 (40). С. 41–52.

67. Секаев В.Г. Использование алгоритмов комбинирования эвристик при построении оптимальных расписаний // Информационные технологии. 2009. № 10. С. 61–64.

68. Семенов С.А. Энергосервис: как достигается эффект // Главный энергетик. 2014. № 11. С. 31–33.

69. Сергиенко Р.Б. Метод формирования нечеткого классификатора самонастраивающимися коэволюционными алгоритмами // Искусственный интеллект и принятие решений. 2010. № 3. С. 98–106.

70. Русина А.Г., Сидоркин Ю.М., Лыкин А.В., Арестова А.Ю., Бородин Д.Н. Оптимизация в электроэнергетических системах: учебно-методическое пособие. – Новосибирск: Изд-во Новосибирского государственного технического ун-та, 2015. 156 с.

71. Скиена С. Алгоритмы. Руководство по разработке: пер. с англ. 2-е изд. – СПб.: БХВ-Петербург, 2013. 720 с.

72. Танаев В.С., Сотсков Ю.И., Струсевич В.А. Теория расписаний. Многостадийные системы. – М.: Наука, Гл. ред. физ.-мат. лит., 1989. 328 с.

73. Токтошов Г.Ы., Монахов О.Г. Об одной модификации алгоритма муравьиной колонии для построения гиперсетей // Труды 12-й международной азиатской школы-семинара «Проблемы оптимизации сложных систем». Новосибирск, 12–16 декабря 2016 г. Новосибирск: Изд-во Новосибирского государственного ун-та, 2016. С. 536–541.

74. Третьякова Е.С., Манусов В.З. Оптимизация реактивной мощности на основе генетического алгоритма // Главный энергетик. 2015. № 1. С. 26–29.

75. Троелсен Э. C# и платформа .NET. Библиотека программиста: пер. с англ. Р. Михеева. – СПб.: Питер, 2004. 796 с.

76. Филиппова Т.А., Сидоркин Ю.М., Русина А.Г. Оптимизация режимов электростанций и энергосистем: учебник. – Новосибирск: Изд-во Новосибирского государственного технического ун-та, 2016. 356 с.

77. Фроловский В.Д. Приближенные методы решения NP-трудных задач в системах автоматизации проектирования: учеб. пособие. – Новосибирск: Изд-во Новосибирского государственного технического ун-та, 2006. 100 с.

78. Фроловский В.Д., Забелин С.Л. Разработка и исследование метаэвристических алгоритмов двумерных матриц и муравьиных колоний для задач геометрического покрытия // Материалы XIV Всероссийской конференции «Математическое программирование и приложения». Екатеринбург, 28 февраля–04 марта 2011 г. Екатеринбург: Институт математики и механики УрО РАН, 2011. С. 174–175.

79. Ходашинский И.А., Бардамова М.Б., Ковалев В.С. (2017a) Построение нечеткого классификатора алгоритмом гравитационного поиска // Доклады Томского государственного университета систем управления и радиоэлектроники. 2017. Т. 20, № 2. С. 84–87.

80. Ходашинский И.А., Самсонов С.С. (2017b) Построение нечеткого классификатора на основе алгоритма обезьян // Бизнес-информатика. 2017. № 1 (39). С. 61–67.

81. Чураков М., Якушев А. Муравьиные алгоритмы [Электронный ресурс]. 2006, 15 с. URL: http://rain.ifmo.ru/cat/data/theory/unsorted/ant-algo-2006/article.pdf (дата обращения: 10.11.2015).

82. Шлее М. Qt 4.8: Профессиональное программирование на C++: пер. с англ. – СПб.: БХВ-Петербург, 2012. 894 с.

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

84. Эхтер Ш., Робертс Дж. Многоядерное программирование: пер. с англ. – СПб.: Питер, 2010. 316 с.

85. Adams J., Balas E., Zawack D. The shifting bottleneck procedure for job shop scheduling // Management Science. 1991. Vol. 34. P. 391–401.

86. Alexandrov D., Kochetov Yu. The Behavior of Ant Colony Algorithm for the Set Covering Problem // Operations Research Proceedings. 1999. P. 255–260.

87. Alrashidi M.R., Hawary M.E. Emission-economic dispatch using a novel constraint handling particle swarm optimization // Canadian Conference on Electrical and Computer Engineering. Ottawa, Canada, May 07–10, 2006. 2006. P 664–669.

88. Anshulika, Bewoor L.A. A genetic algorithm approach for solving a job shop scheduling problem // IEEE International Conference on Computer Communication and Informatics. Coimbatore, India, January 05–07, 2017. 2017. P. 1–6.

89. Artunov V.S., Lisichkin G.V. Energy resources of the 21st century: problems and forecasts. Can renewable energy sources replace fossil fuels? // Russian Chemical Reviews. 2017. Vol. 86. P. 777–804.

90. Barricelli N.A. Esempi numerici di processi di evoluzione // Methodos. 1954. Vol. 6. P. 45–68.

91. Beasley J.E. Operational Research Lirbrary [Электронный ресурс]. URL: http://people.brunel.ac.uk/~mastjjb/jeb/info.html (дата обращения: 11.04.2017).

92. Beck J.C., Fox M.S. Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics // Artificial Intelligence. 2000. Vol. 117, is. 1. P. 31–81.

93. Bellman R. The theory of dynamic programming // Bulletin of the American Mathematical Society. 1954. Vol. 60, is. 6. P. 503–515.

94. Beni G., Wang J. Swarm intelligence in cellular robotic systems // Robots and Biological Systems: Towards a New Bionics. – Germany, Berlin: Springer, 1993. P. 703–712.

95. Bergh F. An Analysis of Particle Swarm Optimizers. PhD thesis [Электронный ресурс]. – RSA, Pretoria: University of Pretoria, 2001. 300 P. URL: https://repository.up.ac.za/bitstream/handle/2263/24297/00thesis.pdf (дата обращения: 25.09.2014).

96. Bertsimas D., Tsitsiklis J. Simulated Annealing // Statistical Sience. 1993. Vol. 8, is. 1. P. 10–15.

97. Biegler L.T., Grossmann I.E. Retrospective on optimization // Computers & Chemical Engineering. 2004. Vol. 28, is. 8. P. 1169–1192.

98. Birbil Ş.İ., Fang S.C. An electromagnetism-like mechanism for global optimization // Journal of global optimization. 2003. Vol. 25, is. 3. P. 263–282.

99. Bishop J.M. Stochastic searching networks // Artificial Neural Networks, First IEEE International Conference (IET). London, UK, October 16–18, 1989. 1989. P. 329–331.

100. Brucker P., Jurisch B., Sievers B. A branch and bound algorithm for the job shop scheduling problem // Discrete Applied Mathematics.1994. Vol. 49. P. 107–127.

101. Brucker P., Knust S. Complex scheduling. – Germany, Berlin: Springer, 2012. 342 p.

102. Carlisle A., Dozier G. An off-the-shelf // Particle Swarm Optimization, Indiana, IN, USA, April 6–7, 2001. 2001. P. 1–6.

103. Chen C., Ye F. Particle swarm optimization algorithm and its application to clustering analysis // IEEE International Conference on Networking, Sensing and Control. Taipei, Taiwan, March 21 –23, 2004. 2004. Vol. 2. P. 789–794.

104. Chvátal V., Klarner D.A., Knuth D.E. Selected combinatorial research problems [Электронный ресурс]. – USA, CA, Stanford: Stanford University, 1972.

31 p. URL: https://pdfs.semanticscholar.org/e832/afd53b720bbb2c3dfff89f5005d5c395719f.pdf (дата обращения: 05.02.2015).

105. Dazahra M.N., Elmariami F., Belfqih A., Boukherouaa J. Optimal Location of SVC using Particle Swarm Optimization and Voltage Stability Indexes // International Journal of Electrical and Computer Engineering. 2016. Vol. 6, is. 6. P. 2581–2588.

106. Dijkstra E.W. A note on two problems in connexion with graphs // Numerische mathematik. 1959. Vol. 1, is. 1. P. 269–271.

107. Dorigo M., Maniezzo V., Colorni A. The Ant System: Optimization by a colony of cooperating agents // IEEE Transactions on Systems, Man, and Cybernetics Part B. 1996. Vol. 26, is. 1. P. 29–41.

108. Dorigo M., Birattari M., Stützle T. Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique [Электронный ресурс]. TR/IRIDIA/2006-023. 2006. 14 p. URL: http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2006-023r001.pdf (дата обращения: 18.09.2014).

109. Eberhart R.C., Kennedy J. New Optimizer Using Particle Swarm Theory // VI International Symposium on Micro Machine and Human Science. Nagoya, Japan, October 4–6, 1995. 1995. P. 39–43.

110. Eberhart R.C., Hu X. Human tremor analysis using particle swarm optimization // Congress on Evolutionary Computation. Washington, DC, USA, Jule 6–9, 1999. 1999. P. 1927–1930.

111. Eberhart R.C., Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization // Congress on Evolutionary Computation (CEC). La Jolla, CA, USA, July 16–19, 2000. 2000. P. 84–88.

112. Eberhart R.C., Shi Y. Particle swarm optimization: developments, applications and resources // Congress on Evolutionary Computation. Seoul, South Korea May 27–30, 2001. 2001. P. 81–86.

113. Edmonds J. Matroids and the greedy algorithm // Mathematical programming. 1971. Vol. 1, is. 1. P. 127–136.

114. Fan H. A modification to particle swarm optimization algorithm // Engineering Computations: International Journal for Computer-Aided Engineering. 2002. Vol. 19. P. 970–989.

115. Filho B., Lima Neto F., Lins A., Nascimento A., Lima M. A novel search algorithm based on fish school behavior // IEEE International Conference on Systems, Man and Cybernetics. Singapore, Singapore, October 12–15, 2008. 2008. P. 2646–2651.

116. Finnila A.B., Gomez M.A, Sebenik C., Stenson C., Doll J.D. Quantum annealing: a new method for minimizing multidimensional functions // Chemical physics letters. 1994. Vol. 219, is. 5–6. P. 343–348.

117. Fisher H., Thompson G. Probabilistic learning combination of local job-shop scheduling rules // Industrial Scheduling. USA, NJ, Englewood: Prentice-Hall, 1963. P. 225–251.

118. Freuder E.C., Mackworth A.K. Constraint satisfaction: An emerging paradigm // Foundations of Artificial Intelligence. 2006. Vol. 2. P. 13–27.

119. Gandomi A.H., Kashani A.R. Evolutionary bound constraint handling for particle swarm optimization // 4th International Symposium on Computational and Business Intelligence (ISCBI). Olten, Switzerland. September 05–07, 2016. 2016. P. 148–152.

120. Garcia-Galan S., Prado R.P., Exposito J.E.M. Rules discovery in fuzzy classifier systems with PSO for scheduling in grid computational infrastructures // Applied Soft Computing. 2015. Vol. 29. P. 424–435.

121. Glover F. Future paths for integer programming and links to artificial intelligence // Computers & operations research. 1986. Vol. 13, is. 5. P. 533–549.

122. Glover F. Tabu search – part I // ORSA Journal on computing. 1989. Vol. 1, is. 3. P. 190–206.

123. Glover F. Tabu search – part II // ORSA Journal on computing. 1990. Vol. 2, is. 1. P. 4–32.

124. Glover F., Laguna M. Tabu Search // Handbook of combinatorial optimization. USA, New York: Springer, 2013. P. 3261–3362.

125. Goss S., Aron S., Deneubourg J.L., Pasteels J.M. Self-organized shortcuts in the Argentine ant // Naturwissenschaften. 1989. Vol. 76, is. 12. P. 579–581.

126. Gromicho J.A.S, Hoorn J.J., Timmer G.T. Exponentially better than brute force: solving the job-shop scheduling problem optimally by dynamic programming // Computers and Operations Research archive. 2012. Vol. 39, is. 12. P. 2968–2977.

127. Harrabi M., Driss O.B., Ghedira K. Combining genetic algorithm and tabu search metaheuristic for job shop scheduling problem with generic time lags // International Conference on Engineering & MIS. Monastir, Tunisia, May 08–09, 2017. 2017. P. 1–8.

128. Heppner F., Grenander U. A stochastic nonlinear model for coordinated bird flocks // The Ubiquity of Chaos. USA, DC, Washington: American Association for the Advancement of Science, 1990. P. 233–238.

129. Hippert H.S., Pedreira C.E., Souza R.C. Neural networks for short-term load forecasting: a review and evaluation // IEEE Transactions on Power Systems. 2001. Vol. 16, is. 1. P. 44–55.

130. Ilie S., Badica C. Multi-Agent Distributed Framework for Swarm Intelligence // Procedia Computer Science. 2013. Vol. 18. P. 611–620.

131. Jamian J.J., Mustafa M.W., Mokhlis H., Baharudin M.A. A New Particle Swarm Optimization Technique in Optimizing Size of Distributed // International Journal of Electrical and Computer Engineering. 2012. Vol. 1, is. 11. P. 137–146.

132. Johnson S.M. Optimal two- and three-stage production schedules with setup times included // Naval Research Logistics Quarterly. 1954. Vol. 1. P. 61–68.

133. Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical report-tr06 [Электронный ресурс]. – Turkey, Kayseri: Erciyes University, 2005. 10 p. URL: http://www.dmi.unict.it/mpavone/nc-cs/materiale/tr06_2005.pdf (дата обращения: 18.03.2016).

134. Keller J.M., Liu D., Fogel D.B. Collective Intelligence and Other Extensions of Evolutionary Computation. – USA, NJ, Hoboken: John Wiley & Sons, Inc., 2016. 400 p.

135. Kennedy J., Eberhart R.C. Particle swarm optimization // International Conference on Neural Networks (ICNN'95). Perth, WA, Australia, 21 Nov. – 1 Dec., 1995. 1995. P. 1942–1948.

136. Kirkpatrick S., Gelatt Jr., Vecchi M.P. Optimization by Simulated Annealing // Science. 1983. Vol. 220, is. 4598. P. 671–680.

137. Kuhn H.W., Tucker A.W. Nonlinear programming [Электронный ресурс] // Traces and emergence of nonlinear programming. Switzerland, Basel: Birkhäuser, 2014. P. 247–258. URL: http://web.math.ku.dk/~moller/undervisning/MASO2010/kuhntucker 1950.pdf (дата обращения: 10.11.2017).

138. Land A.H., Doig A.G. An automatic method of solving discrete programming problems // Econometrica: Journal of the Econometric Society. 1960. Vol. 28, is. 3. P. 497–520.

139. Lawler E.L., Lenstra J.K, Rinnooy Kan A.H.G., Shmoys D.B. Sequencing and Scheduling: algorithms and complexity. – Eindhoven: Technische Universiteit Eindhoven, 1989. 70 p.

140. Lee T.-Y. Operating schedule of battery energy storage system in a time-of-use rate industrial user with wind turbine generators: a multipass iteration particle swarm optimization approach // IEEE Transactions on Energy Conversion. 2007. Vol. 22, is. 3. P. 774–782.

141. Levenberg K. A method for the solution of certain non-linear problems in least squares // Quarterly of applied mathematics. 1944. Vol. 2, is. 2. P. 164–168.

142. Liu B., Wang L., Jin Y.-H. An effective PSO-based memetic algorithm for flow shop scheduling // IEEE Transactions on Systems, Man, and Cybernetics. 2007. Part B: Cybernetics. Vol. 37, is. 1. P. 18–27.

143. Mahadevan K., Kannan P.S. Comprehensive learning particle swarm optimization for reactive power dispatch // Applied Soft Computing. 2010. Vol. 10. P. 641–652.

144. Malek M. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem // Annals of Operations Research. 1989. Vol. 21, is. 1. P. 59–84.

145. Manusov V.Z., Matrenin P.V. Optimization of Fuzzy Controller of a Wind Power Plant Based on the Swarm Intelligence // XIII International scientific-technical conference on actual problems of electronic instrument engineering (APEIE). Novosibirsk, Russia, October 03–06, 2016. 2016. Vol. 1, part 4. P. 293–298.

146. Manusov V., Matrenin P., Kokin S. (2017a) Swarm intelligence algorithms for the problem of the optimal placement and operation control of reactive power sources into power grids // International Journal of Design & Nature and Ecodynamics. 2017. Vol. 12, is. 1. P. 101–112.

147. Marquardt D.W. An algorithm for least-squares estimation of nonlinear parameters // Journal of the society for Industrial and Applied Mathematics. 1963. Vol. 11, is. 2. P. 431–441.

148. Metropolis N., Rosenbluth A.W., Rosenbluth M.N., Teller A.H., Teller E. Equation of state calculations by fast computing machines // The journal of chemical physics. 1953. Vol. 21, is. 6. P. 1087–1092.

149. Passino K.M. Biomimicry of bacterial foraging for distributed optimization and control // IEEE control systems. 2002. Vol. 22, is. 3. P. 52–67.

150. Pham D.T., Ghanbarzadeh A., Koç E., Otri S., Rahim S., Zaidi M. Bee Algorithm, A Novel Approach to Function Optimisation [Электронный ресурс]. 2005. 40 p. URL: https://www.researchgate.net/publication/260985621_The_Bees_Algorithm_Technical_Note (дата обращения: 02.10.2014).

151. Pedersen M., Chipperfield A.J. (2010a) Simplifying Particle Swarm Optimization // Applied Soft Computing. 2010. Vol. 10, is. 2. P. 618–628.

152. Pedersen M. (2010b) Good Parameters for Particle Swarm Optimization [Электронный ресурс]. Hvass Laboratories, 2010. 12 p. URL: http://www.hvass-labs.org/people/magnus/publications/pedersen10good-pso.pdf (дата обращения: 05.11.2015).

153. Pezzella F., Merelli E. A tabu search method guided by shifting bottleneck for the job shop scheduling problem // European Journal of Operational Research. 2000. Vol. 120. P. 297–310.

154. Poli R. An Analysis of Publications on Particle Swarm Optimisation Applications. Department of Computer Science [Электронный ресурс]. – UK, Essex: University of Essex, 2007. 57 p. URL: http://dces.essex.ac.uk/staff/poli/technical-reports/tr-csm-469.pdf (дата обращения: 27.02.2015).

155. Puchta M. Optimirung von Problemstellungen aus der diskreten und der Prozess – Industrie unter Vewendung physikalischer Verfahren [Электронный ресурс]. – Austria, Reginsbugr: Physik der Universität Regensburg, 2004. 246 p. URL: https://epub.uni-regensburg.de/10242/1/Dissertation.pdf (дата обращения: 10.12.2017).

156. Quevedo J., Cazakevicius F.E., Beltrame R.C. Analysis and Design of an Electronic On-Load Tap Changer Distribution Transformer for Automatic Voltage Regulation // IEEE Transactions on Industrial Electronics. 2017. Vol. 64, is. 1. P. 883–894.

157. Rabanal P., Rodríguez I., Rubio F. Using River Formation Dynamics to Design Heuristic Algorithms [Электронный ресурс]. 2007. 15 p. URL: https://pdfs.semanticscholar.org/202d/0d60c5b3110ce922af742b818e2bc60d45b6.pdf (дата обращения: 16.08.2017).

158. Rado R. A theorem on independence relations // The Quarterly Journal of Mathematics. 1942. Vol. 1. P. 83–89.

159. Rashedi E., Nezamabadi-Pour H., Saryazdi S. GSA: a gravitational search algorithm // Information science. 2009. Vol. 179, is. 13. P. 2232–2248.

160. Reynolds C. Flocks, Herds, and Schools: A Distributed Behavioral Model // Computer Graphics. 1987. Vol. 21, is. 4. P. 25–34.

161. Rinnooy Kan A.H.G., Magazine M.J. Report of the session on scheduling // Ann. Discr. Math. 1975. Vol. 5. P. 423–426.

162. Robbins B.A., Zhu H., Domínguez-García A.D. Optimal Tap Setting of Voltage Regulation Transformers in Unbalanced Distribution Systems // IEEE Transactions on Power Systems. 2016. Vol. 31, is. 1. P. 256–267.

163. Schrijver A. On the history of combinatorial optimization (till 1960) // Handbooks in operations research and management science. 2005. Vol. 12. P. 1–68.

164. Shah-Hosseini H. Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem // International Journal of Intelligent Computing and Cybernetics. 2008. Vol. 1, is. 2. P. 193–212.

165. Shi Y., Eberhart R. (1998a) A modified particle swarm optimizer // IEEE International Conference on Evolutionary Computation. Anchorage, AK, USA, May 04–09, 1998. 1998. P. 69–73.

166. Shi Y., Eberhart R. (1998b) Parameter selection in particle swarm optimization // VII International Conference on Evolutionary Programming Evolutionary Programming (EP98). San Diego, CA, USA, March 25–27, 1998. 1998. P. 591–600.

167. Shi Y., Eberhart R. Empirical study of particle swarm optimization // Congress on Evolutionary Computation (CEC99). Washington, DC, USA, Jule 06–09, 1999. 1999. P. 1945–1950.

168. Spatti D.H., Silva I.N., Usida W.F., Flauzino R.A. Fuzzy Control System for Voltage Regulation In Power Transformers // IEEE Latin America Transactions. 2010. Vol. 8, is. 1. P. 51–57.

169. Stützle T., Hoos H. MAX-MIN Ant System and local search for the traveling salesman problem // IEEE International Conference on Evolutionary Computation (ICEC 97). Indianopolis, IN, USA, April 13 –16, 1997. 1997. P. 309–314.

170. Thangaraj R., Pant M., Abraham A., Bouvry P. Particle swarm optimization: Hybridization perspectives and experimental illustrations // Applied Mathematics and Computation. 2011. Vol. 217. P. 5208–5226.

171. Weise T. Global optimization algorithms: Theory and application [Электронный ресурс]. 2011. 1223 p. URL: http://www.it-weise.de/projects/bookNew.pdf (дата обращения: 22.07.2017).

172. Trelea I.C. The Particle Swarm Optimization Algorithm: convergence analysis and parameter selection // Information Processing Letters. 2003. Vol. 85. P. 317–325.

173. Valle Y., Venayagamoorthy G.K., Mohagheghi S., Hernandes J.-C., Harley R.G. Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power

Systems // IEEE Transactions on Evolutionary Computation. 2008. Vol. 12, is. 2. P. 171–195.

174. Wolpert D.H., Macready W.G. No Free Lunch Theorems for Optimization // IEEE Transactions on Evolutionary Computation. 1997. Vol. 1, is. 67. P. 67–82.

175. Wolpert D.H. What the no free lunch theorems really mean [Электронний ресурс]. 2013. 15 p. URL: https://dl.acm.org/ft_gateway.cfm?id=2555237&type=pdf (дата обращения: 30.03.2018).

176. Wu H.-Sh., Zhang F.-M. Wolf Pack Algorithm for Unconstrained Global Optimization [Электронний ресурс] // Mathematical Problems in Engineering. 2014. 17 p. URL: https://www.hindawi.com/journals/mpe/2014/465082 (дата обращения: 20.08.2017).

177. Yang X. (2010a). A new metaheuristic bat-inspired algorithm // Nature inspired cooperative strategies for optimization. – Germany, Berlin: Springer. 2010. P. 65–74.

178. Yang X. (2010b). Firefly algorithm, Stochastic Test Function and Design Optimization // International Journal of Bio-Inspired Computation. 2010. Vol. 2, is. 2. P. 78–84.

179. Yoshida H., Kawata K., Fukuyama Y., Takayama Sh. A particle swarm optimization for reactive power and voltage control considering voltage security assessment // IEEE Transactions on Power Systems. 2000. Vol. 15, is. 4. P. 1232–1239.

180. Zhang T., Yu C., Zhang Y., Tian W. Ant Colony System Based on the ASRank and MMAS for the VRPSPD // International Conference om Wireless Communications, Networking and Mobile Computing. Shanghai, China, September 21–25, 2007. 2007. P. 3728–3731.

181. Zhao R., Tang W. Monkey algorithm for global numerical optimization // Journal of Uncertain Systems. 2008. Vol. 2, is. 3. P. 165–176.

182. Zhao D., Luo L., Zhang K. An improved ant colony optimization for the communication network routing problem // Mathematical and Computer Modelling. 2010. Vol. 52, is. 11–12. P. 1976–1981.

# Приложение А

## Акты о внедрении результатов работы

УТВЕРЖДАЮ

Генеральных директор

АО «Уральский электрохимический комбинат»

_____А.А. Белоусов

«05» 11 2015 г.

АКТ ВНЕДРЕНИЯ

результатов диссертационной работы Матренина П.В. «Разработка адаптивных алгоритмов роевого интеллекта в проектировании и управлении техническими системами»

Настоящим подтверждаю, что результаты диссертационных исследований Матренина П.В. «Разработка адаптивных алгоритмов роевого интеллекта в проектировании и управлении техническими системами» обладают актуальностью и представляют практический интерес для систем электроснабжения предприятий. По результатам технического совещания в АО «Уральский электрохимический комбинат» рассмотрена перспективность предложенного в отчете Новосибирского государственного технического университета «О научно-исследовательской работе по оптимизации размещения установок компенсации реактивной мощности в сетях электроснабжения 0,4 кВ» (авторы: Е.С. Третьякова, П.В. Матренин) подхода к снижению потерь активной мощности с помощью глубокой компенсации и выбору размещения компенсирующих установок на основании алгоритма роя частиц. В отчете дано описание составленной математической модели, примененного метода оптимизации и показано, что глубокая компенсация может снизить потери электроэнергии рассматриваемой подстанции на 22% со сроком окупаемости 3 года для нерегулируемых установок компенсации реактивной мощности и 3.5 года для регулируемых установок.

Предложено рекомендовать рассмотрение разработанного и описанного в диссертационных работах подхода для применения на подстанциях комбината при проведении работы по их реконструкции и техническому перевооружению.

Главный энергетик АО «УЭХК» _____ В.Р. Лекомцев

Копия верна

Начальник Отдела кадров НГТУ О.К. Пустовалова

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ

Проректор по научной работе НГТУ

_____ А.Г.Вострецов

«___» _____ 2015 г.

# ОТЧЕТ
# О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ ПО ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ УСТАНОВОК КОМПЕНСАЦИИ РЕАКТИВНОЙ МОЩНОСТИ В СЕТЯХ 0,4 кВ ЭЛЕКТРОСНАБЖЕНИЯ АО «УРАЛЬСКИЙ ЭЛЕКТРОХИМИЧЕСКИЙ КОМБИНАТ»

Научный руководитель работы

Докт. технических наук, профессор _____ В.З.Манусов

Новосибирск 2015

Копия верна

Начальник Отдела кадров НГТУ К. Пуховалова

УТВЕРЖДАЮ:

Проректор по учебной работе Новосибирского государственного технического университета, доктор технических наук, доцент

С.В. Брованов

«14» мая 2018 г.

# АКТ ВНЕДРЕНИЯ

результатов диссертационной работы Матренина П.В. «Разработка адаптивных алгоритмов роевого интеллекта в проектировании и управлении техническими системами»

Настоящим актом подтверждается, что результаты диссертационной работы Матренина П.В. «Разработка адаптивных алгоритмов роевого интеллекта в проектировании и управлении техническими системами» на соискание ученой степени кандидата технических наук по специальности 05.13.01 «Системный анализ, управление и обработка информации», используются в учебном процессе на кафедре «Системы электроснабжения предприятий» Новосибирского государственного технического университета. Разработанные в диссертационной работе адаптивные алгоритмы роевого интеллекта для решения задач оптимизации и управления техническим системами излагаются в дисциплинах «Интеллектуальные системы электроснабжения», «Оптимизация систем электроснабжения» и «Системный анализ в электроэнергетике». Также в них используются созданные программные приложения визуализации стохастических оптимизационных алгоритмов. Изданное в НГТУ учебное пособие П.В. Матренина «Методы стохастической оптимизации» и материалы диссертационной работы используются при написании бакалаврских и магистерских диссертаций, а также в исследованиях аспирантов.

Павлюченко Дмитрий Анатольевич
кандидат технических наук, доцент,
кафедра «Системы электроснабжения предприятий», заведующий кафедрой

Копия верна
О.К. Пустовалова
Отдела кадров НГТУ

# Приложение Б
# Свидетельства на программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

## СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2012614741

Адаптивная система календарного планирования методом муравьиной колонии

Правообладатель(ли): *Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования Новосибирский государственный технический университет (НГТУ) (RU)*

Автор(ы): *Матренин Павел Викторович, Секаев Виктор Гилячевич (RU)*

Заявка № 2012612776

Дата поступления 11 апреля 2012 г.

Зарегистрировано в Реестре программ для ЭВМ *28 мая 2012 г.*

*Руководитель Федеральной службы по интеллектуальной собственности*

*Б.П. Симонов*

Копия верна

Начальник Отдела кадров НГТУ Пустовалова

РОССИЙСКАЯ ФЕДЕРАЦИЯ

# СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2013661938

Система календарного планирования адаптивными алгоритмами роевого интеллекта

Правообладатель(ли): *федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Новосибирский государственный технический университет" (RU)*

Автор(ы): *Матренин Павел Викторович (RU), Секаев Виктор Гилячевич (RU)*

Заявка № 2013660015
Дата поступления 31 октября 2013 г.
Зарегистрировано в Реестре программ для ЭВМ
*19 декабря 2013 г.*

*Руководитель Федеральной службы по интеллектуальной собственности*

*Б.П. Симонов*

Копия верна
О.К. Пустовалова
Отдела кадров КГТУ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

# СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

## № 2015613847

«Система визуализации стохастических алгоритмов оптимизации»

Правообладатель: *Матренин Павел Викторович (RU)*

Автор: *Матренин Павел Викторович (RU)*

Заявка № **2015610644**
Дата поступления **09 февраля 2015 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***26 марта 2015 г.***

*Врио руководителя Федеральной службы по интеллектуальной собственности*

*Л.Л. Кирий*

Копия верна

О.К. Пустовалова
Отдел кадров НГТУ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

# СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

**№ 2018611639**

**«Программа «ПРОГНОЗИРОВАНИЕ» для построения моделей графиков электропотребления роевыми алгоритмами»**

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Новосибирский государственный технический университет» (RU)***

Авторы: ***Матренин Павел Викторович (RU), Худжасаидов Джахонгир Худжасаидович (TJ), Русина Анастасия Георгиевна (RU)***

Заявка № **2017663020**

Дата поступления **13 декабря 2017 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***02 февраля 2018 г.***

*Руководитель Федеральной службы по интеллектуальной собственности*

*Г.П. Ивлиев*

Копия верна

О.К. Лустовалова

Начальник Отдела кадров КГТУ

# Приложение В

## Программный интерфейс библиотеки роевых алгоритмов

```
#ifndef I_TASK_H
#define I_TASK_H

#include <vector>

class I_Task
{
public:
   virtual double calc_criterion(const std::vector<double>& parameter_vector) = 0;
   virtual int get_dimension() const = 0;
   virtual ~I_Task() {}
};

class I_Combinatorial_Task
{
public:
   virtual double calc_criterion(const std::vector<int>& parameter_vector) = 0;
   virtual int get_dimension() const = 0;
   virtual std::vector<int> get_variant_vector() const noexcept = 0;
   virtual ~I_Combinatorial_Task() {}
};

#endif // I_TASK_H



#ifndef I_SWARM_H
#define I_SWARM_H

#include <vector>
#include <memory>
#include <algorithm>
```

```
#include "random_generator.h"

class I_Task;

struct Swarm_Unit
{
   Swarm_Unit(int dimension)
   {
      m_position.resize(dimension);
      for (double& p : m_position)
      {
         p = Random_Generator::get_instance().get_random_double(0., 1.);
      }
   }

   Swarm_Unit(const Swarm_Unit& copy)
      : m_position(copy.m_position)
      , m_criterion(copy.m_criterion)
   {

   }

   std::vector<double> m_position;
   double m_criterion;
};



class I_Swarm
{
protected:
   std::vector<double> m_global_best_position;
   std::shared_ptr<I_Task> m_ptr_task;
   double m_best_criterion;
   unsigned int m_iteration_number;
```

```
    unsigned int m_swarm_size;
    bool m_is_trace = false;

protected:
    virtual void criterion_calculation() = 0;
    virtual void movements() = 0;

public:
    I_Swarm(std::shared_ptr<I_Task> ptr_task, unsigned int iteration_number
                                                    , unsigned int swarm_size)
    {
        m_ptr_task = ptr_task;
        m_iteration_number = iteration_number;
        m_swarm_size = swarm_size;
    }

    virtual unsigned int get_parameter_number() const = 0;
    virtual void set_parameters_0_1(const std::vector<double> parameter_vector) = 0;
    virtual void full_reset() = 0;
    virtual void reset() = 0;
    virtual void light_reset() = 0;
    virtual double run()
    {
        for (int iteration = 0; iteration < m_iteration_number; ++iteration)
        {
            criterion_calculation();
            movements();
        }

        return m_best_criterion;
    }

    virtual double run(const std::vector<double>& parameter_vector) = 0;

    virtual double get_best(std::vector<double>* solution_vector) const
```

```
    {
        solution_vector->clear();
        solution_vector->insert(solution_vector->begin(), m_global_best_position.begin(),
                                                          m_global_best_position.end());
        return m_best_criterion;
    }
    virtual void set_trace(bool trace)
    {
        m_is_trace = trace;
    }
    virtual ~I_Swarm() {}
};

#endif // I_SWARM_H
```

# Приложение Г

## Размерности задач календарного планирования Operational Research Library

| Задача | Число требований | Число приборов | Задача | Число требований | Число приборов |
|---|---|---|---|---|---|
| Abz5 | 10 | 10 | La19 | 10 | 10 |
| Abz6 | 10 | 10 | La20 | 10 | 10 |
| Abz7 | 15 | 20 | La21 | 15 | 10 |
| Abz8 | 15 | 20 | La22 | 15 | 10 |
| Abz9 | 15 | 20 | La23 | 15 | 10 |
| Ft06 | 6 | 6 | La24 | 15 | 10 |
| Ft10 | 10 | 10 | La25 | 15 | 10 |
| Ft20 | 20 | 5 | La26 | 20 | 10 |
| La01 | 10 | 5 | La27 | 20 | 10 |
| La02 | 10 | 5 | La28 | 20 | 10 |
| La03 | 10 | 5 | La29 | 20 | 10 |
| La04 | 10 | 5 | La30 | 20 | 10 |
| La05 | 10 | 5 | La31 | 30 | 10 |
| La06 | 15 | 5 | La32 | 30 | 10 |
| La07 | 15 | 5 | La33 | 30 | 10 |
| La08 | 15 | 5 | La34 | 30 | 10 |
| La09 | 15 | 5 | La35 | 30 | 10 |
| La10 | 15 | 5 | La36 | 30 | 10 |
| La11 | 20 | 5 | La37 | 30 | 10 |
| La12 | 20 | 5 | La38 | 30 | 10 |
| La13 | 20 | 5 | La39 | 30 | 10 |
| La14 | 20 | 5 | La40 | 15 | 15 |
| La15 | 20 | 5 | Yn1 | 20 | 20 |
| La16 | 10 | 10 | Yn2 | 20 | 20 |
| La17 | 10 | 10 | Yn3 | 20 | 20 |
| La18 | 10 | 10 | Yn4 | 20 | 20 |