Федеральное государственное бюджетное образовательное учреждение высшего образования «Липецкий государственный педагогический университет имени П.П. Семенова-Тян-Шанского»

На правах рукописи

СКАКОВ Евгений Сергеевич

МЕТОДЫ И АЛГОРИТМЫ ИНТЕЛЛЕКТУАЛЬНОЙ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПО ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ ЭЛЕМЕНТОВ РАЗВИВАЮЩИХСЯ ИНФОРМАЦИОННЫХ СИСТЕМ

Специальность:
05.13.01 – Системный анализ, управление и обработка информации (информационные и технические системы)

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
Малыш Владимир Николаевич

Липецк – 2017

Оглавление

ВВЕДЕНИЕ

Актуальность темы. В современном мире все большую роль играют информационные системы (ИС). Подобные системы предоставляют своим пользователям вычислительные ресурсы, услуги по обработке, поиску и хранению информации, отвечают за обеспечение глобального мультимедийного общения. Важнейшим требованием, предъявляемым к информационным системам, является обеспечение качественного, эффективного и надежного сервиса обслуживания клиентов.

В ближайшем будущем основным фактором, влияющим на развитие информационных систем, станет рост числа пользователей и, соответственно, растущие требования к производительности систем и качеству предоставляемых ими услуг. Важным аспектом является тот факт, что все решения по изменению структуры информационной системы (добавление элементов, модернизация элементов, исключение элементов из системы) имеют долгосрочные последствия. Данный факт обуславливают важность принятия оптимальных решений на стадиях планирования и оптимизации информационных систем.

Стоит отметить, что большинство работ, посвященных задачам размещения, направлено на решение задач данного типа на стадии предварительного планирования создаваемых систем. Однако для современных информационных систем более актуальной является проблема оптимизации уже существующей инфраструктуры системы.

Также среди недостатков большинства существующих работ, посвященных рассматриваемой проблеме можно отметить: решение методами, не показывающими высокую скорость расчета в задачах большой размерности; математическая модель не подразумевает использование нескольких типов элементов; неуниверсальность математической модели – работы, как правило, рассматривают конкретный тип информационных систем (например,

беспроводные), конструируя математическую модель, не применимую для других типов.

Учитывая вышеизложенное, актуальность темы диссертационного исследования продиктована необходимостью разработки алгоритмического обеспечения систем интеллектуальной поддержки принятия решений по оптимизации размещения элементов информационных систем в условиях их развития. Она подразумевает создание методов и алгоритмов, позволяющих находить близкое к оптимальному решение NP-трудных задач данного типа за приемлемое время. Также актуальной является задача разработки алгоритмического обеспечения процесса настройки управляющих параметров метаэвристических алгоритмов решения задачи размещения элементов развивающихся информационных систем, позволяющей значительно улучшать их производительность.

Теоретическую основу исследования составляют современные научные работы, посвященные проблемам планирования и оптимизации информационных систем таких исследователей, как В.М. Вишневский, В.О. Тихвинский, С.Ю. Ермолаев, В.Л. Бурковский, M. St-Hilaire, E. Amaldi. Среди работ, посвященных метаэвристическим алгоритмам оптимизации, наиболее значимыми являются труды Ю.А. Кочетова, J.H. Holland, F. Glover, S. Kirkpatrick, M. Dorigo, D. Karaboga, D. Teodorovic. Вопросам оптимизации значений управляющих параметров для метаэвристических алгоритмов посвящены труды А.П. Карпенко, A.E. Eiben, S.K. Smit.

Работа выполнена в ФГБОУ ВО «Липецкий государственный педагогический университет имени П.П. Семенова-Тян-Шанского» в рамках научного направления «Исследование методов идентификации и оптимизации производительности компьютерных сетей».

Цель и задачи исследования. Цель работы заключается в разработке методов и алгоритмов интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем.

Для достижения поставленной цели в диссертационной работе необходимо решить следующие **задачи**:

– провести системный анализ проблематики интеллектуальной поддержки принятия решений по оптимизации структуры информационных систем;

– предложить математическую модель задачи размещения элементов развивающихся информационных систем;

– разработать метаэвристические алгоритмы решения задачи размещения элементов развивающихся информационных систем;

– провести системный анализ процесса настройки управляющих параметров для разработанных метаэвристик;

– разработать метод оптимизации значений управляющих параметров для разработанных метаэвристик;

– разработать программную реализацию системы интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем;

– провести вычислительный эксперимент на основе созданного программного обеспечения с целью обоснования эффективности предложенных алгоритмов и методов.

Методы исследования. В качестве теоретической и методологической основы диссертационного исследования использованы методы системного анализа, оптимизации, теория графов, метаэвристические методы, технологии структурного и объектно-ориентированного программирования.

Тематика работы соответствует следующим пунктам паспорта специальности 05.13.01: п. 4 «Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации», п. 5 «Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений и обработки информации», п. 10 «Методы и алгоритмы

интеллектуальной поддержки при принятии управленческих решений в технических системах».

Научная новизна. В работе получены следующие результаты, отличающиеся научной новизной.

1. Математическая модель задачи размещения элементов развивающихся информационных систем, отличающаяся учетом возможности модернизации элементов системы, свойств мест-кандидатов, помех при межмодульном взаимодействии элементов системы и подразумевающая использование нескольких типов элементов информационной системы.

2. Алгоритмы интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем, основанные на популяционных метаэвристиках и метаэвристиках локального поиска, позволяющие находить близкие к оптимальным решения NP-трудных задач данного типа за приемлемое время. Отличительной особенностью предлагаемых алгоритмов является: для алгоритмов локального поиска – построение окрестности решения при помощи небольших операций, вносящих изменения в конфигурацию информационной системы; для муравьиного алгоритма – использование двух колоний искусственных муравьев для двухфазного решения задачи.

3. Метод оптимизации управляющих параметров разработанных метаэвристических алгоритмов на базе эволюционного подхода, отличающийся возможностью одновременной настройки вещественных, целочисленных и символьных параметров и обеспечивающий значительное улучшение производительности алгоритмов решения задачи размещения элементов развивающихся информационных систем.

4. Структура программного комплекса интеллектуальной поддержки принятия решений, реализующего методы и алгоритмы оптимизации размещения элементов развивающихся информационных систем, отличительной особенностью которой является наличие инструментов автоматизации проведения

серий вычислительных экспериментов с целью оценки качества алгоритмов, лежащих в основе созданной программной системы.

Практическая значимость. Использование результатов данного диссертационного исследования при проведении работ по оптимизации и планированию развивающихся информационных систем способствует их успешному и качественному выполнению в части, относящейся к выбору мест для размещения элементов системы и определению состава оборудования системы, что позволяет уменьшить капитальные затраты на инфраструктуру системы и обеспечить требуемые параметры качества обслуживания для пользователей.

Был разработан программный комплекс интеллектуальной поддержки принятия решений по оптимизации и планированию беспроводных сетей передачи данных, который может быть использован телекоммуникационными компаниями и проектными организациями при оптимизации и планировании беспроводных информационных систем.

На элементы программных средств получены свидетельства о государственной регистрации.

Реализация и внедрение результатов работы. Основные теоретические и практические результаты, полученные в диссертационном исследовании, были внедрены на объектах ОАО "Мобильные ТелеСистемы" (филиал в Липецкой области) и АО "ЭР-Телеком Холдинг" (филиал в г. Липецк) при проведении работ по развитию радиосетей стандарта 3G/4G, что подтверждается соответствующими актами.

Основные результаты работы внедрены в учебный процесс Липецкого государственного педагогического университета имени П.П. Семенова-Тян-Шанского в рамках дисциплин «Методы теории принятия решений», «Методы оптимизации», при выполнении курсового и дипломного проектирования, что подтверждается соответствующим актом.

Апробация результатов. Основные положения диссертационной работы докладывались и обсуждались на следующих конференциях: XV-й

Международной научно-практической конференции «Новое слово в науке и практике: гипотезы и апробация результатов исследований» (Новосибирск, 2015); Международной научно-практической конференции «Информационные управляющие системы и технологии» (Одесса, Украина, 2014–2016); III-й Международной конференции "Устойчивость и процессы управления" (Санкт-Петербург, 2015); V-й Международной научно-практической конференции «Актуальные проблемы технических наук» (Уфа, 2015); V-й Международной научно-практической конференции «Актуальные проблемы технических наук» (Тамбов, 2015); XVIII-й Международной научно-технической конференции «Проблемы передачи и обработки информации в сетях и системах телекоммуникаций» (Рязань, 2015); Международном симпозиуме «Надежность и качество» (Пенза, 2016); Международной летней научной школе «Парадигма» (Варна, Болгария, 2015).

Публикации. По теме исследования опубликовано 23 работы, отражающих основные положения исследования, среди которых 6 статей в журналах, рекомендованных ВАК РФ; 1 статья в издании, индексируемом Scopus; 3 свидетельства о государственной регистрации программ для ЭВМ.

В работах, опубликованных в соавторстве, автору принадлежат: в [57,50] – постановка и формализация задачи размещения элементов развивающихся информационных систем (на примере беспроводной сети передачи данных) как частного случая NP-трудной задачи размещения с ограничениями на мощности; [55,51,156] – алгоритм размещения элементов развивающихся информационных систем на базе эволюционного подхода; [44,156] – алгоритм имитации отжига для решения задачи размещения элементов развивающихся информационных систем; [46] – алгоритмы поиска с запретами и мультистарта для решения задачи размещения элементов развивающихся информационных систем; [53] – пчелиный алгоритм BCO для решения задачи размещения элементов развивающихся информационных систем; [155,47] – муравьиный алгоритм для решения задачи размещения элементов развивающихся информационных систем; [54] –

муравьиный алгоритм решения задачи размещения центров обработки данных при проектировании распределенной информационной системы; [48] – пчелиный алгоритм ABC для решения задачи размещения элементов развивающихся информационных систем; [45] – проведение вычислительного эксперимента; [49] – муравьиный алгоритм для решения задачи размещения с ограничениями на мощности; [29,52] – основанный на эволюционном подходе метод настройки параметров метаэвристик решения задачи размещения элементов развивающихся информационных систем (на примере беспроводной сети передачи данных).

Структура и объем работы. Диссертационная работа состоит из введения, четырёх глав, заключения, приложений, списка использованных источников, включающего 175 наименований. Основная часть работы изложена на 168 страницах и содержит 20 рисунков и 12 таблиц.

1 ПОСТАНОВКА ЗАДАЧИ РАЗМЕЩЕНИЯ ЭЛЕМЕНТОВ РАЗВИВАЮЩИХСЯ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1 Системный анализ проблематики интеллектуальной поддержки принятия решений при планировании и оптимизации информационных систем

1.1.1 Общие сведения об информационных системах

Для начала дадим определение информационной системы (ИС). Заранее отметим, что в научной литературе не существует единого установившегося определения ИС, поэтому для получения наиболее полного представления о сущности информационных систем ниже будет приведено сразу несколько определений ИС.

Согласно [64], информационная система – это система, предназначенная для сбора, хранения, обработки и поиска информации, необходимой для обеспечения процессов управления, проектирования и т.п., а также для удовлетворения потребностей индивидуального потребителя информации.

В широком смысле информационную систему можно определить как любое хранилище информации (архивы, библиотеки, картотеки, наборы статистических данных и т.п. [64,3]).

Информационная система – система, предназначенная для удовлетворения индивидуальных потребностей потребителя информации [3].

Информационная система – это совокупность программного, технического, организационного обеспечения и персонала, предназначенная для своевременного обеспечения людей информацией [95].

Федеральный закон РФ от 27 июля 2006 года № 149-ФЗ «Об информации, информационных технологиях и о защите информации» определяет ИС как совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств [66].

ИС – совокупность вычислительного и коммуникационное оборудования, ПО, информационных ресурсов, персонала, обеспечивающая поддержку динамической информационной модели некоторой части реального мира для удовлетворения информационных потребностей пользователей [23].

Существует несколько классификаций информационных систем [21].

1) По степени распределенности:

– настольные;

– распределенные.

2) По степени автоматизации:

– автоматизированные;

– автоматические.

3) По решаемым задачам:

– справочные;

– информационно-поисковые;

– расчетные;

– технологические.

4) По масштабности задач:

– персональные;

– групповые;

– корпоративные.

5) По сфере применения:

– финансовые;

– офисные;

– медицинские и т.д.

Объектом исследования данной диссертационной работы являются современные развивающиеся информационные системы. Наибольший акцент будет сделан на современные сети передачи данных (в т.ч. беспроводные).

Элемент системы – это предел членения системы с точки зрения аспекта ее рассмотрения, решения конкретной задачи, поставленной цели [64]. В данной

работе под элементами (узлами) информационных систем будет, в первую очередь, подразумеваться оборудование, из которого ИС состоит.

Данное диссертационное исследование посвящено проблематике размещения элементов развивающихся информационных систем, т.е. систем с некоторой уже реализованной инфраструктурой, которые нуждаются в модернизации в условиях нарастающей конкуренции.

Возможные причины необходимости внесения изменений в текущую конфигурацию информационной системы:

– изменение потребностей клиентов, которые уже подключены к ИС;

– изменение числа пользователей на территории, которая уже охвачена информационной системой;

– изменение количества услуг, предоставляемых системой;

– желание владельца ИС расширить территорию влияния информационной системы;

– изменение состава имеющегося в наличии оборудования.

Стоит отметить, что, как правило, вносить изменения в структуру системы приходится из-за совокупности вышеприведенных факторов.

Изменения в ИС осуществляются при помощи следующих операций:

– установка нового узла (элемента) на свободное место-кандидат;

– расформирование (удаление, деинсталляция) существующего узла, т.е. освобождение места-кандидата;

– модернизация (апгрейд, обновление) существующего элемента системы;

– подключение пользователя в систему;

– удаление пользователя из системы;

– удаление связей между элементами системы, между клиентами и элементами.

1.1.2 Информационные системы с точки зрения системного анализа

Если рассматривать типичную современную информационную систему, то с точки зрения классификации систем [63,69,3] её можно назвать:

– открытой, т.к. она способна обмениваться информацией и энергией с внешней средой;

– материальной, т.к. она существует в объективной реальности;

– искусственной, т.к. она создана человеком;

– технической, т.к. в ее основе лежат устройства и детали;

– многоэлементной (в большинстве случаев);

– гетерогенной, т.к. она состоит из разнородных элементов (в большинстве случаев);

– иерархической, т.к. ее элементы располагаются на разных уровнях иерархии, состоящих в отношениях подчинения и выполняющих разные функции (в большинстве случаев);

– динамической, т.к. она характеризуется функционированием и развитием во времени.

На сегодняшний день одним из основных подходов к решению задач моделирования и оптимизации информационных и технических процессов и систем является системный анализ, который можно определить как «метод научного познания, который состоит в том, что любой объект по отношению к субъекту рассматривается как система (сложное образование), состоящая из большого числа элементов, связанных между собой вещественными, энергетическими, информационными и другими связями сильнее, чем с окружающей средой» [10,13].

Согласно [10], основными характеристиками материальной технической системы являются её функциональное назначение, структура, компоновка, организация и совокупность показателей качества. Раскроем суть этих категорий для информационных систем.

1) **Функциональным назначением** информационной системы является предоставление пользователям услуг. Под услугами в ИС понимается доступ к информации. Причем подразумевается предоставление услуг определенного заранее заданного качества за счет покрытия заранее заданной территории.

2) **Структура –** некоторая организация системы посредством синтеза из отдельных элементов, обладающих определенными свойствами и характеризующих цель и назначение системы. Структура отражает качественный и количественный состав, множество связей между элементами и определяет основные свойства системы [10].

Приведем в качестве примера описание структуры такой ИС, как беспроводная сеть передачи данных. Элементами структуры беспроводной информационной сети широкополосного доступа являются:

А) Оборудование поставщика телекоммуникационных услуг:

– базовые станции;

– коммутационное, шлюзовое и прочее вспомогательное оборудование.

Свойства оборудования:

– технические характеристики (мощность сигнала, пропускная способность и т.д.).

– способность поддерживать определенные сетевые протоколы и стандарты.

Б) Клиенты.

Свойства клиентов:

– местоположение клиентского оборудования;

– потребности клиентов в телекоммуникационных услугах.

Стоит отметить, что, несмотря на то, что клиентское оборудование является частью беспроводной сети передачи данных, операторы связи при построении сетей занимаются решением задачи проектирования и оптимизации топологии только той части системы, которая составлена из их оборудования. Можно сказать, что свойства клиентов сети являются входными данными для задач проектирования и оптимизации беспроводных сетей передачи данных.

Взаимодействие базовых станций и клиентов осуществляется:

А) В нисходящем направлении (downlink). Передача информации от базовой станции к клиенту.

Б) В восходящем направлении (uplink). Передача информации от клиентского оборудования к базовой станции.

Внешняя среда такой ИС, как беспроводная сеть передачи данных, состоит из двух частей:

А) Среда распространения сигнала. В случае беспроводных информационных сетей средой распространения сигнала, каналом передачи информационных потоков между устройствами и беспроводной инфраструктурой, является воздух.

Б) Сети, которые являются внешними для беспроводной сети. Это, в первую очередь:

– Интернет;

– телефонные сети общего пользования.

3) **Компоновка** характеризуется геометрическим размещением элементов системы в заданном пространстве [10]. Как правило, под компоновкой подразумевается расположение оборудования, составляющего комплекс технических средств системы. Например, для беспроводной сети передачи данных местоположения пользователей услуг являются входными данными для задач проектирования и оптимизации сети, а вот местоположение оборудования и устройств (базовые станции, контроллеры и т.д.) выбирается исходя из критериев оптимальности и условий допустимости конфигурации системы.

В значительной степени данное диссертационное исследование направлено на решение задачи оптимизации компоновки рассматриваемых систем, т.е. ЗРЭРИС – задача компоновки.

4) **Организация** системы подразумевает актуализацию и упорядочение связей между элементами системы [10]. Характерным примером описания организации информационных систем являются сетевые стандарты и протоколы,

которые описывают полный перечень связей между элементами таких информационных систем, как сети передачи данных, и специфику подобных взаимодействий.

5) **Вектор показателей эффективности** системы задается совокупностью качественных и количественных показателей, характеризующих полезный эффект от использования системы [10].

Так как данная работа посвящена разработке алгоритмов интеллектуальной поддержки принятия решений по размещению элементов информационных систем, применимых как на стадии предварительного планирования ИС, так и на этапе модернизации, то необходимо рассмотреть критерии качества, характеризующих ИС на любом этапе ее жизненного цикла. Раздел 1.1.7 посвящен обзору научной литературы, имеющему цель выявить основные подходы к формулировке критериев оптимальности и допустимости структуры информационных систем при решении задачи оптимизации размещения их элементов.

1.1.3 Задача размещения элементов развивающихся информационных систем с точки зрения системного анализа

Существуют три основные задачи, относящиеся к изучению и созданию систем: анализ, синтез и принятие решений [42].

Анализ состоит в изучении свойств и поведения систем (и их элементов) в различных условиях функционирования. При проектировании развивающихся информационных систем анализ заключается в следующем:

– изучение особенностей протоколов и стандартов, лежащих в основе построения проектируемой информационной системы;

– изучение особенностей отдельных доступных для установки элементов ИС (оборудования);

– изучение особенностей взаимодействия между элементами сети;

– изучение особенностей внешней среды;

– изучение особенностей взаимодействия элементов системы и внешней среды;

– изучение потребностей пользователей;

– изучение особенностей уже существующей (исходной) ИС, которую необходимо модернизировать.

Синтез состоит в построении возможных вариантов систем. Он состоит в построении структуры системы (определении элементов проектируемой системы и связей, возникающих между ними), а также в определении параметров системы и ее элементов при фиксированной структуре [10]. При проектировании информационных систем синтез заключается в построении множества всех возможных вариантов системы на основе результатов анализа.

Принятие решений состоит в выборе наилучшего варианта системы из нескольких альтернативных и оценке эффективности его функционирования. Функционирование системы характеризуется некоторым количественным или качественным функционалом, который называют показателем эффективности [10].

Как известно, лицо, принимающее решение, (ЛПР) – это человек или коллектив, который на этапе принятия решений осуществляет выбор и несет ответственность за его последствия [64].

Решение задачи размещения элементов развивающихся информационных систем относится к фазам синтеза и принятия решений, т.к. лицо, принимающее решения, на этапе синтеза формирует множество допустимых вариантов модернизированной информационной системы (за счет отсева вариантов, не удовлетворяющих требованиям, предъявляемым к системе), которые определяются свойствами мест-кандидатов, свойствами клиентов, свойствами внешней среды, свойствами оборудования, которое имеется в распоряжении). Далее на этапе принятия решений ЛПР должно выбрать наилучшую альтернативу с точки зрения минимизации или максимизации некоторого критерия (или критериев) качества системы.

В данной работе предлагается подход, согласно которому задача принятия решения сводится к задаче математического дискретного программирования, где ограничения отвечают за проверку решения (конфигурации комплекса оборудования) на допустимость, а целевая функция – за учет критериев качества системы. При таком подходе ЛПР должно сформулировать критерии допустимости решений и целевую функцию задачи дискретного программирования (см. рисунок 1.1). Получившуюся задачу требуется решить при помощи алгоритмов интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем.

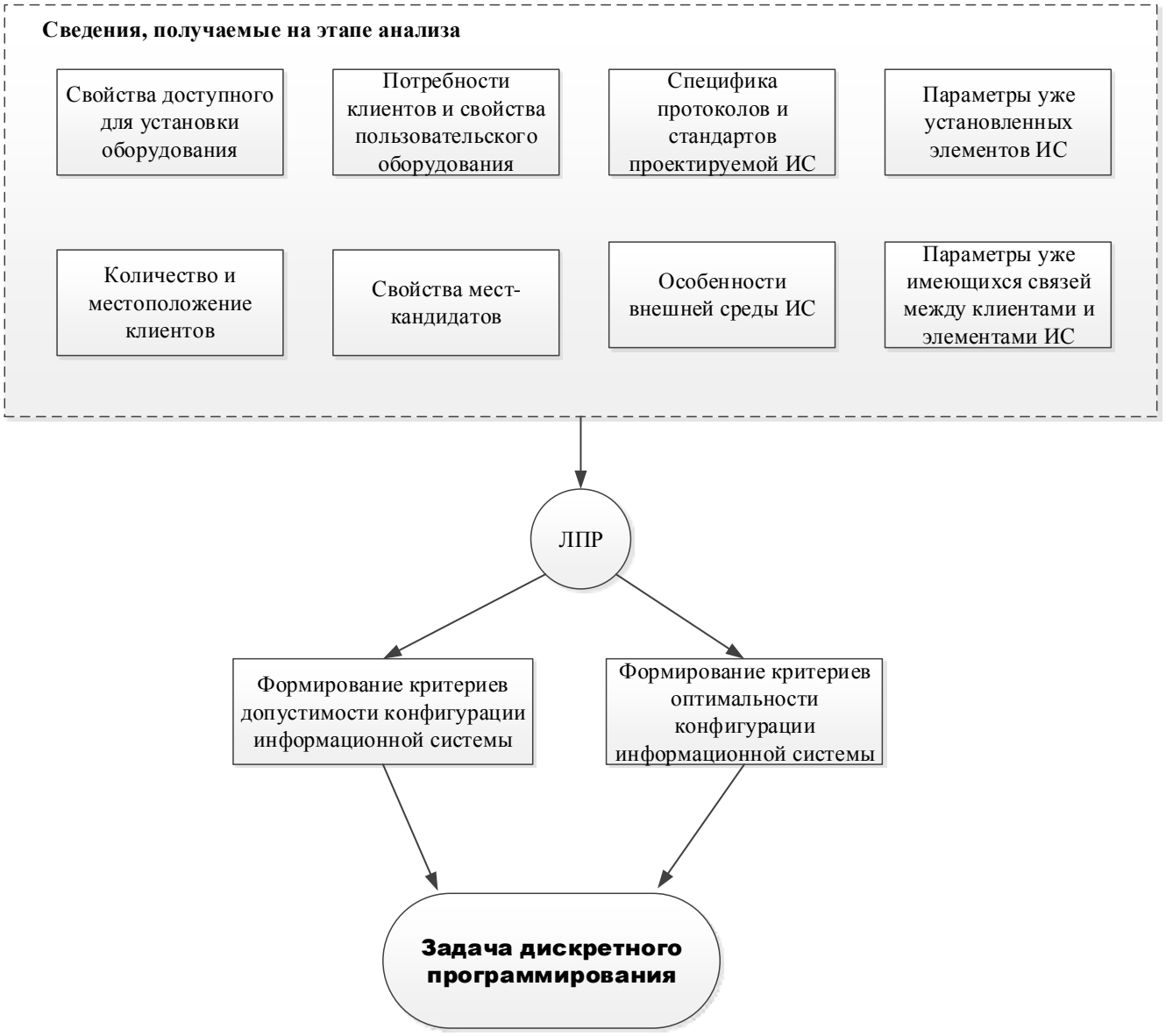

Рисунок 1.1 – Структура процесса принятия решений по оптимизации размещения элементов развивающихся информационных систем

1.1.4 Этапы создания информационной системы

Выделяют следующие этапы построения коммерческих информационных систем ([18]):

1. Формирование целей планирования ИС, оценка необходимых инвестиций, времени реализации и окупаемости.

2. Анализ требований потенциальных клиентов (пользователей).

3. Формирование топологии системы, определение мест размещения элементов системы.

4. Строительство системы.

5. Настройка параметров качества функционирования системы.

6. Сдача ИС в эксплуатацию.

7. Предоставление услуг клиентам на платной основе.

8. Мониторинг статистических параметров функционирования ИС.

9. Осуществление процесса оптимизации ИС на основе результатов мониторинга (в идеале непрерывно происходящий процесс).

Задача, рассматриваемая в данной диссертационной работе, относится к третьему и девятому этапам построения ИС. На третьем этапе, относящемся к фазе предварительного планирования, решается задача достижения единовременного экономического эффекта за счет снижения затрат на закупку оборудования при реализации решений, касающихся топологии системы [18]. Дополнительный экономический выигрыш может быть получен за счет того, что корректно проведенные работы на этапе предварительного планирования позволят обеспечить предоставление клиентам услуг высокого качества за счет полного обеспечения требований клиентов к вычислительным и информационным ресурсам. А это, в свою очередь, способно увеличить постоянные доходы владельца ИС. На девятом этапе решается аналогичная задача с той лишь разницей, что входными данными для подобной задачи являются параметры (места, где установлено оборудование, тип установленного

оборудования и т.д.) уже существующей развивающейся информационной системы. Цели оптимизации систем и их предварительного планирования одинаковые: обеспечение высокого уровня качества обслуживания пользователей при минимально возможных денежных вложениях.

В определенном смысле можно сказать, что проектирование ИС «с нуля» – частный случай модернизации ИС, при котором осуществляется модернизация системы, не имеющей в своем составе ни одного элемента. Именно поэтому исследование и решение задач, входящих в комплекс проблем по оптимизации развивающихся информационных систем, имеет большую теоретическую и практическую значимость.

Стоит отметить, что после этапа первоначальной сдачи ИС в эксплуатацию начинается «жизнь» развивающейся информационной системы. При этом «жизнь» развивающейся информационной системы носит циклический характер (см. рисунок 1.2).

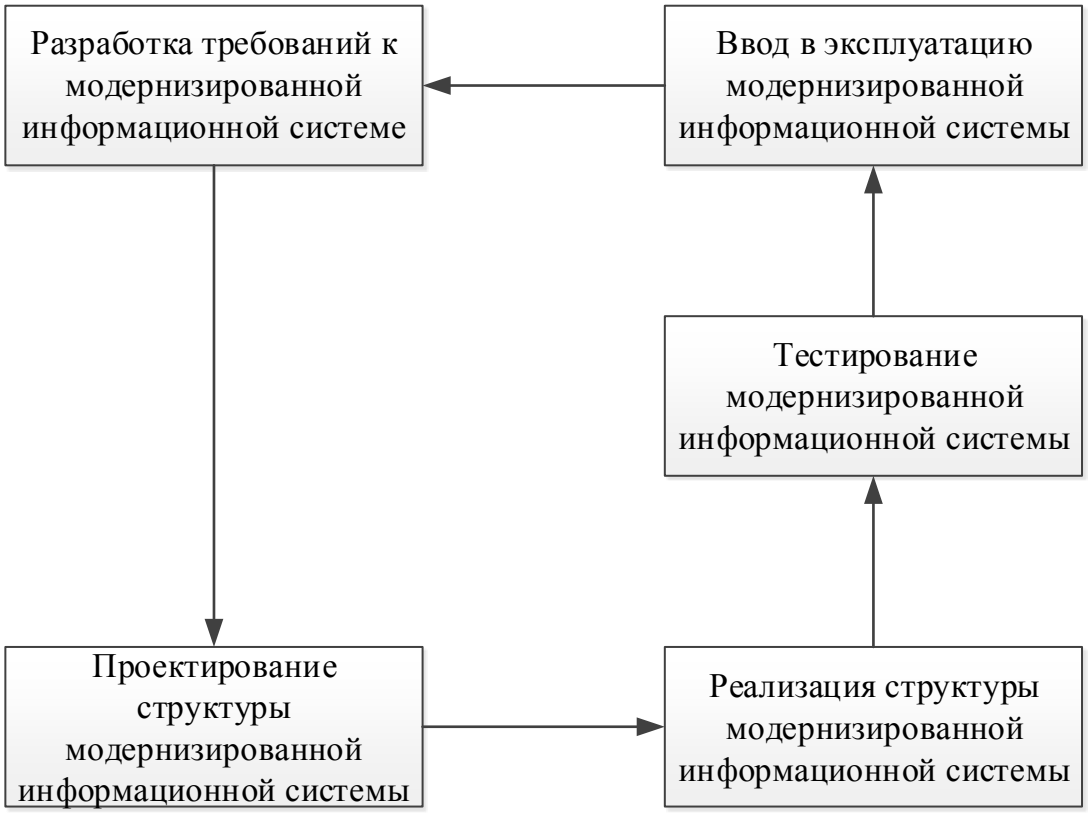

Рисунок 1.2 – Жизненный цикл развивающейся информационной системы

1.1.5 Технология NGN

На сегодняшний день наиболее распространенными информационными системами являются:

– телефонные информационные системы;

– телевизионные информационные системы;

– компьютерные информационные системы;

– радиосистемы.

Стоит отметить, что данные типы систем могут пересекаться. Например, телефонная связь может осуществляться по беспроводному принципу, т.е. сеть оператора мобильной связи является одновременно и телефонной ИС, и радиосистемой.

Современные ИС разного типа могут быть объединены в информационную систему более высокого уровня в рамках технологии NGN (Next Generation Networks, сети следующего поколения) [128].

В основе NGN лежит концепция объединения информационных систем разного типа в единую сеть. Сети NGN основаны на протоколе IP и технологии MPLS (MultiProtocol Label Switching, многопротокольная коммутация по меткам).

Основные характеристики NGN-систем [146]:

– пакетная передача данных;

– конвергения мобильных услуг с услугами, предоставляемыми сетями фиксированного доступа;

– функции разделяются на транспортные функции, функции управления вызовами и функции управления услугами;

– независимость функций, связанных с услугами, от базовых транспортных технологий;

– взаимодействие пользователей с сетями через открытые интерфейсы;

– поддержка широкополосных технологий со сквозным (end-to-end) обеспечением QoS (Quality of Service, качество обслуживания);

– соответствие всем нормативным требованиям, например, касающимся аварийной связи, безопасности/конфиденциальности и т.д.;

– неограниченный доступ клиентов к различным поставщикам услуг;

– поддержка широкого спектра технологий при реализации сети доступа.

Сеть NGN имеет два уровня [88]:

– уровень услуг, отвечающий за передачу пользовательских данных, а также функции управления, необходимые для предоставления услуг;

– транспортный уровень, отвечающий за реализацию функций управления и поддержки транспортных ресурсов для передачи данных между устройствами системы.

В таблице 1.1 приведено соответствие уровней сетей NGN и уровней базовой эталонной модели взаимодействия открытых систем (сетевой модели OSI) [128].

Таблица 1.1 – Соответствие уровней NGN и сетевой модели OSI

Уровни NGN	**Уровни модели OSI**
уровень услуг	прикладной
	уровень представления
	сеансовый
	транспортный
транспортный	сетевой
	канальный
	физический

Пример сети следующего поколения приведен на рисунке 1.3 [88]. Поясним основные элементы, приведенные на рисунке [88,146,128].

– Шлюз доступа (Access gateway, AG), который позволяет подключать абонентские лини к пакетной сети; преобразует потоки трафика аналогового доступа в пакеты; обеспечивает абонентский доступ к сети и услугам NGN.

– Транспортный (медиа) шлюз (Media Gateway, MG). Отвечает за функции преобразования речевой информации из формата TDM (Time Division

Multiplexing, мультиплексирование с разделением по времени) в IP-пакеты и маршрутизацию IP-пакетов.

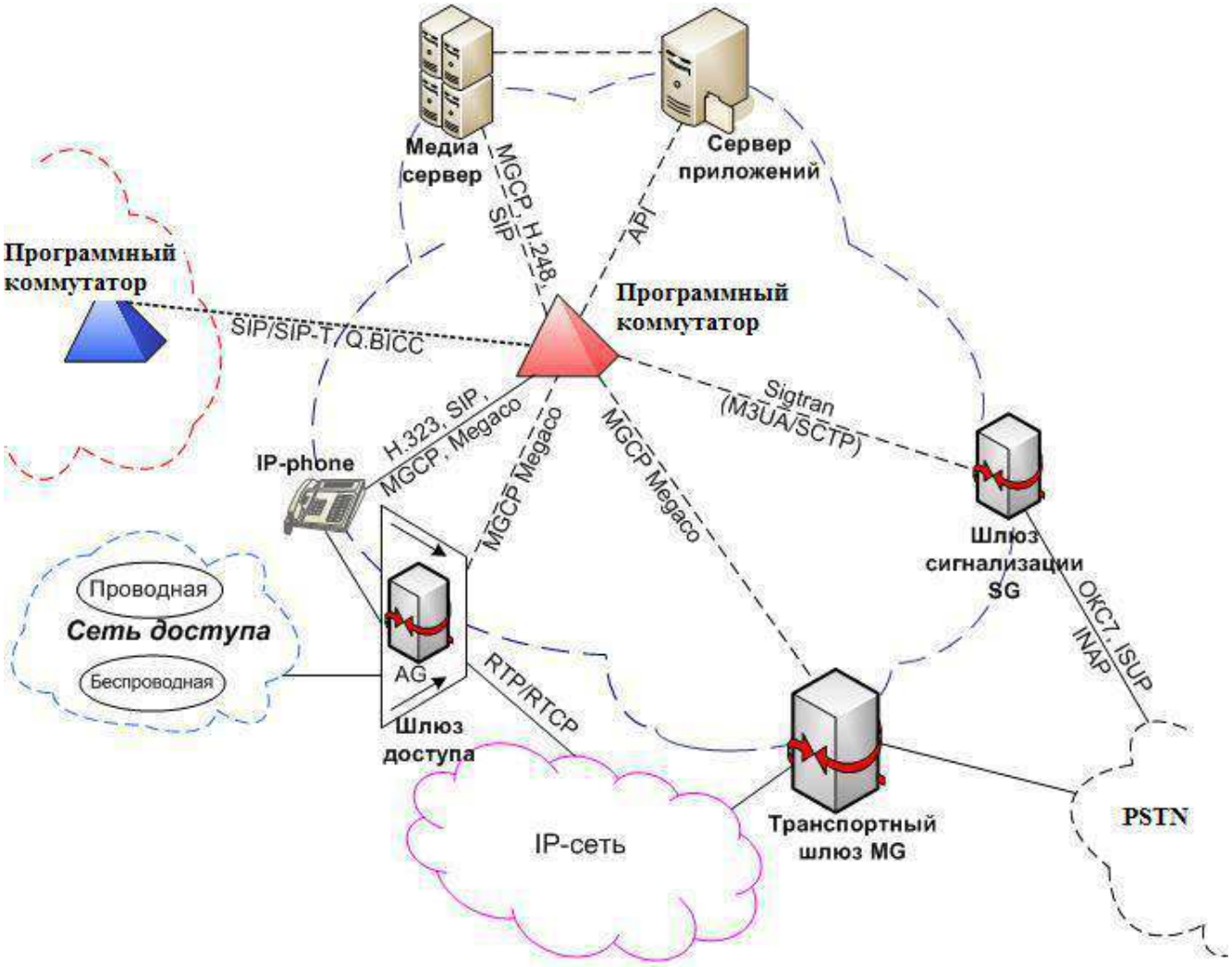

Рисунок 1.3 – Пример сети NGN

– Шлюз сигнализации (Signalling Gateway, SG). Блок, который обеспечивает преобразование сигналов между NGN и другими сетями (например, осуществляет преобразование STP в SS7).

– Программный коммутатор (Softswitch). Обеспечивает «контроль доставки услуг» в сети; отвечает за управление вызовами и управление шлюзами через протокол H.248; может выполнять функции шлюза сигнализации для взаимодействия с сетью сигнализации PSTN (Public Switched Telephone Network, телефонная сеть общего доступа).

– Сервер приложений (Application Server, AS). Блок, который поддерживает выполнение различных служб, например, служб управления вызовом и специальные ресурсы NGN (например, медиа-сервер, message-сервер).

1.1.6 Межмодульное взаимодействие элементов системы

Отличительной особенностью современных информационных систем, помимо тенденций к их непрерывному развитию и улучшению, является необходимость учитывать аспекты межмодульного взаимодействия [27,67]. Межмодульное взаимодействие отражает факт взаимного влияния друг на друга процессов, происходящих с разными элементами системы. В зависимости от конкретной реализации ИС, специфика межмодульного взаимодействия и вызываемых им помех может отличаться. Например, если рассматривать беспроводную сеть передачи данных как частный случай развивающейся информационной системы, а базовую станцию как частный случай элемента ИС, то в качестве помех при межмодульном взаимодействии стоит рассматривать межсотовые помехи (подробнее см. раздел 1.5).

Таким образом, процессы, происходящие, с одним элементом ИС, могут серьезно влиять на способность другого элемента удовлетворять потребности пользователей.

1.1.7 Обзор современных подходов к процессу принятия решений по оптимизации размещения элементов при планировании и оптимизации информационных систем

Как уже было сказано, процесс принятия решений при проектировании некоторой системы подразумевает выбор лучшего варианта из нескольких альтернативных на основании некоторого критерия (показателя). Набор вариантов, среди которых осуществляется выбор, формируется на этапе синтеза.

Основным подходом к оптимизация развивающихся информационных систем, в частности к оптимизации расположения отдельных элементов ИС, распространённым в научной литературе, является представление задачи размещения элементов системы в виде оптимизационной задачи математического программирования. В такой задаче ограничения отвечают за отсечение недопустимых вариантов конфигурации системы (что соответствует этапу синтеза), а целевая функция формулирует некий критерий качества, на основании значения которого осуществляется выбор лучшего из решений (что соответствует этапу принятия решений).

Работы [1,2] В.М. Вишневского и его коллег – одни из первых работ, посвященных решению задачи оптимального размещения элементов (базовых станций) при проектировании беспроводных информационных систем. Среди других русскоязычных публикаций по данной проблеме отметим работы С.Ю. Ермолаева [17,18], посвященные сетям WiMAX. Работы M. St-Hilaire [160,161] посвящены проблеме планирования топологии сетей стандарта UMTS. Отметим работы В.Л. Бурковского и его коллег [14,15,16,30,31,32], в которых решается более универсальная проблема проектирования и оптимизации телекоммуникационной сети, которую можно свести к ЗРЭРИС. Статьи E. Amaldi и его соавторов [75,77,74,76] посвящены проектированию беспроводных информационных систем стандарта 3G. Работа [132] направлена на решение задачи размещения элементов ИС в сетях стандарта LTE при помощи аппарата теории игр.

Стоит отметить, что большинство работ по рассматриваемой тематике (кроме работ В.Л. Бурковского и его коллег [14,15,16,30,31,32]) рассматривает исключительно проблему проетирования информационных систем «с нуля», игнорируя вопросы развития структуры уже существующих ИС.

Критерии допустимости решений задачи размещения элементов развивающихся информационных систем

Далее будут приведены основные подходы к формулировке ограничений задачи математического программирования, направленной на оптимизацию элементов информационных систем, представленные в научной литературе.

1) Ограничение уникальности. Только один элемент ИС может быть установлен на одном месте, пригодном для установки (т.н. месте-кандидате). Очевидное ограничение, присутствующее во всех работах по данной тематике.

2) Необходимость подключения всех клиентов (пользователей). Данное ограничение присутствует в работах В.М. Вишневского [1,2], С.Ю. Ермолаева [17,18], M. St-Hilaire [160,161]. В работах Д.Э. Елизарова и В.Л. Бурковского [14,15,16] подобное ограничение отсутствует. Вместо него введено ограничение на соблюдение планового показателя минимальной суммарной потребности пользователей в услугах телекоммуникационной сети. Также подобное ограничение отсутствует в математической модели в работах E. Amaldi [75,77,74,76], однако в них целевая функция направлена на максимизацию суммарного трафика, предоставляемого пользователям.

3) Обязательное выполнение ограничения (превышение) суммарного итогового показателя ценности V_{min} для всех размещенных объектов системы. [14,15,16]. В качестве V_{min}, например, может выступать необходимый суммарный трафик.

4) Ограничение на производительность оборудования. Данное ограничение присутствует в большинстве работ по рассматриваемой тематике.

5) Ограничения на допустимость подключения клиента к элементу ИС. Важно понимать, что подобные ограничения специфичны для каждого типа информационных систем (проводные, беспроводные и т.д.).

Например, в беспроводных сетях передачи данных причина, по которой подключение может быть невозможным, заключается в том, что вследствие затухания радиосигнал от передатчика к приемнику может приходить слишком

слабым для нормальной работы, т.е. дело в территориальной удаленности между элементами информационной системы и клиентами. В работах [1,2,17,18], подобные ограничения задаются при помощи фиксированного радиуса действия каждой базовой станции.

Согласно исследованию [33] пользователь должен находиться в зоне влияния некоторого телекоммуникационного узла, чтобы подключение к нему было возможно. При этом критерии нахождения в зоне влияния явно не сформулированы и должны быть предложены ЛПР для каждого конкретного варианта реализации информационноц системы.

Критерии оптимальности решений задачи размещения элементов развивающихся информационных систем

Далее будут перечислены основные подходы к формулировке целевой функции задачи математического программирования, направленной на оптимизацию размещения элементов информационных систем, представленные в научной литературе.

1) Минимизируемая суммарная стоимость оборудования и его монтажа. Данный оптимизационный критерий используется в большинстве работ по рассматриваемой тематике, причем в большинстве из них он собственно и представляет собой целевую функцию задачи. Отметим, что в работах [14,15,16] решается задача оптимизации и развития уже существующей инфраструктуры, а потому из суммарной стоимости закупаемого оборудования вычитается стоимость оборудования, выводимого из эксплуатации.

2) В работах E. Amaldi [75,77,74,76] одним из слагаемых целевой функции выступает суммарный трафик, предоставляемый клиентам, который необходимо максимизировать.

3) Работа [18], рассматривающая беспроводную сеть передачи данных, в качестве одного из слагаемых минимизируемой целевой функции включает потери при распространении сигнала. Стоит заметить, что наиболее корректно

было бы учитывать потери при распространении сигнала в ограничениях задачи, делая невозможным подключение при слабом уровне сигнала, чего в работе [18] не сделано.

4) Модель, предложенная в работах [30,31,32,33], учитывает затраты на предоставление услуг пользователям. Однако в работах [14,15,16] справедливо замечено, что ими можно пренебречь, т.к. издержки на обеспечение работы с клиентами и предоставление им необходимых сервисов являются статистически предсказуемыми и, как следствие, не подлежат оптимизации в рамках модели, представляющей собой задачу дискретного программирования.

Таким образом, на данный момент в научной литературе не сформирован единый подход к решению задачи размещения элементов информационных систем. Также наблюдается недостаток работ, направленных на сравнительный анализ эффективности решения задачи проектирования сетевой инфраструктуры ИС при помощи широкого спектра разнообразных метаэвристик.

1.2 Постановка цели и задач работы

Цель работы заключается в разработке методов и алгоритмов интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем.

Для достижения поставленной цели в диссертационной работе необходимо решить следующие **задачи**:

– провести системный анализ проблематики интеллектуальной поддержки принятия решений по оптимизации структуры информационных систем;

– предложить математическую модель задачи размещения элементов развивающихся информационных систем;

– разработать метаэвристические алгоритмы решения задачи размещения элементов развивающихся информационных систем;

– провести системный анализ процесса настройки управляющих параметров для разработанных метаэвристик;

– разработать метод оптимизации значений управляющих параметров для разработанных метаэвристик;

– разработать программную реализацию системы интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем;

– провести вычислительный эксперимент на основе созданного программного обеспечения с целью обоснования эффективности предложенных алгоритмов и методов.

1.3 Задачи размещения

1.3.1 Классификация задач размещения

Задачи размещения – широкий пласт математических моделей исследования операций, интересный как с практической точки зрения, так и с точки зрения комбинаторной оптимизации [25]. Данный класс задач находит применение при решении проблем размещения предприятий, складов и магазинов, планировании метро, размещении узлов сетей связи, станций обслуживания и др. В качестве целевой функции, как правило, используются либо величина затрат на создание и функционирование системы технических средств, либо суммарная эффективность системы (например, объем выполняемых работ) [25].

Задачи размещения делятся на два типа:

– задачи размещения без ограничения на емкость источника обслуживания (uncapacitated facility location problem, UFLP);

– задачи размещения с ограничением на емкость источника обслуживания (capacitated facility location problem, CFLP).

Задачи CFLP, в свою очередь, подразделяются на два подтипа:

– CFLP с одним источником обслуживания (single-source capacitated facility location problem, SSCFLP);

– CFLP с несколькими источниками обслуживания (multi-source capacitated facility location problem, MSCFLP).

Отметим, что задачи размещения являются примером задач комбинаторной оптимизации (т.е. их решение представляет собой комбинацию уникальных компонент, выбираемых из конечного набора; цель – поиск оптимальной комбинации компонент).

1.3.2 Простейшая задача размещения

Опишем простейшую задачу размещения [25,39]. Имеется множество пунктов возможного размещения предприятий $I - \{1, \ldots, m\}$. Имеется множество клиентов $J - \{1, \ldots, n\}$. Предприятия могут производить некоторый продукт в неограниченном количестве. Известны стоимости размещения предприятий c_i в указанных пунктах и затраты t_{ij} на транспортировку продукции от предприятия в пункте i потребителю j ($i \in I$, $j \in J$). Требуется разместить предприятия и прикрепить к ним клиентов так, чтобы суммарные производственно-транспортные затраты были минимальны [28]. С использованием введенных обозначений оптимизационная постановка задачи может быть записана следующим образом:

$$F(y, X) = \sum_{i \in I} c_i y_i + \sum_{i \in I} \sum_{j \in J} t_{ij} x_{ij} \to \min, \tag{1.1}$$

$$\sum_{i \in I} x_{ij} = 1, \quad y_i \geq x_{ij}, \quad i \in I, \quad j \in J, \tag{1.2}$$

$$y_i \in \{0,1\}, \quad x_{ij} \geq 0, \quad i \in I, \quad j \in J, \tag{1.3}$$

где x_{ij} – доля спроса клиента j, которую обеспечивает предприятие из пункта i; $y_i = 1$, если предприятие открыто в пункте i, 0 – в противном случае ($i \in I, j \in J$).

Первое слагаемое (1.1) отражает затраты на открытие предприятий, второе определяет производственно-транспортные расходы. Сформулированная задача известна как простейшая задача размещения производства. Она является обобщением известной задачи о покрытии множествами и, следовательно, относится к числу NP-трудных в сильном смысле [39,129,9].

1.3.3 Задача размещения с ограничениями на мощности

Задача размещения с ограничениями на мощности предприятий является обобщением простейшей задачи размещения. В отличие от последней, в ней предполагается, что каждое предприятие может производить продукцию только в ограниченных количествах [19]. Подобное предположение меняет математическую модель и значительно усложняет методы решения оптимизационной задачи.

В задаче размещения с ограничениями на мощности известна мощность производства V_i для каждого $i \in I$, а также потребность d_{ij} каждого клиента j в продукции предприятия i ($i \in I$, $j \in J$) [28]. Оптимизационная постановка задачи может быть записана следующим образом:

$$F(y, X) = \sum_{i \in I} c_i y_i + \sum_{i \in I} \sum_{j \in J} t_{ij} x_{ij} \to \min, \tag{1.4}$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{1.5}$$

$$\sum_{j \in J} d_{ij} x_{ij} \le V_i y_i, \quad i \in I, \tag{1.6}$$

$$y_i, x_{ij} \in \{0, 1\}, \quad i \in I, \quad j \in J, \tag{1.7}$$

Минимизируемая целевая функция (1.4) имеет смысл суммарных затрат на открытие предприятий и обслуживание потребителей. Ограничение (1.5) требует удовлетворения потребностей всех потребителей, причем одного потребителя может обслуживать только одно предприятие. Ограничение (1.6) несет сразу две функции. Оно позволяет обслуживать потребителей только с открытых предприятий, а также ограничивает сверху возможные объемы поставок продукции с каждого предприятия. Таким образом, задача размещения с ограничениями на мощности относится к классу SSCFLP и является NP-трудной [19]. Если же переменная x_{ij} не является целочисленной, а ее значение лежит в диапазоне [0; 1], то такая задача относится к классу MSCFLP и также является NP-трудной [19].

В работах M.S. Daskin [92,91,93] приведена подробная классификация и история наиболее распространенных классических задач размещения.

1.4 Задача размещения элементов развивающихся информационных систем

1.4.1 Постановка задачи размещения элементов развивающихся информационных систем

Согласно Вишневскому [2], задача проектирования сети сводится к отысканию минимума функционала приведенной стоимости

$$C(U,\Omega,Y) \to \min \tag{1.8}$$

при наличии ограничений на вероятностно-временные и структурные характеристики сети

$$V_i(U,\Omega,Y) \le V_{i0} \tag{1.9}$$

и требовании принадлежности множества вариантов архитектуры сети $Q(U,\Omega,Y)$ к области технически реализуемых решений

$$Q(U,\Omega,Y) \in Q_0. \tag{1.10}$$

Здесь U – векторная величина, отражающая параметры сетевой нагрузки; Ω – векторная величина, представляющая собой совокупность параметров технических средств; Y – векторная величина, отражающая параметры логической структуры сети.

Данный подход можно обобщить, применив его к процессу проектирования распределенной ИС, исходя из того, что всякая сеть – частный случай информационной системы. Более того, в данном исследовании будет рассмотрена более широкая проблема – проблема развития существующей инфраструкты информационной системы.

Задача размещения элементов развивающихся информационных систем (ЗРЭРИС) заключается в том, что имеется ИС некой конфигурации (назовем её исходной ИС). Возникает потребность в модернизации системы. Есть некоторое множество клиентов (N_{cl} штук). Модернизированная ИС должна удовлетворить потребности всех пользователей путем подключения их к своим элементам (узлам). Есть заранее определенное множество потенциальных мест (N_{ps} штук),

куда могут быть установлены элементы ИС (т.н. места-кандидаты). Имеется N_{types} типов доступных для установки элементов ИС, отличающихся по своим характеристикам. Задача сводится к минимизации затрат по переходу от исходного состояния ИС к модернизированному при выполнении ряда ограничений.

Введем следующие обозначения:

T – множество различных доступных для установки типов элементов ИС;

I – множество клиентов;

S – множество мест-кандидатов;

$y^0_{st} = 1$, если элемент типа $t \in T$ установлен в исходной ИС на месте $s \in S$, иначе – 0;

$x^0_{is} = 1$, если в исходной ИС клиент $i \in I$ подсоединен к элементу, установленному на месте-кандидате $s \in S$, иначе – 0;

$y_{st} = 1$, если элемент типа $t \in T$ установлен в модернизированной ИС на месте $s \in S$, иначе – 0;

$x_{is} = 1$, если в модернизированной ИС клиент $i \in I$ подсоединен к элементу, установленному на месте-кандидате $s \in S$, иначе – 0;

prf_t – максимальная производительность узла типа $t \in T$;

b_i – потребности клиента $i \in I$.

Требование уникальности – в каждом месте-кандидате может быть установлен только один элемент информационной системы::

$$\sum_{t \in T} y_{st} \le 1 \qquad \forall s \in S. \tag{1.11}$$

Каждый клиент должен быть подключен только к одному элементу ИС, причем не должно остаться неподключенных клиентов:

$$\sum_{s \in S} x_{is} = 1 \qquad \forall i \in I. \tag{1.12}$$

Клиент может быть «отнесен» к месту-кандидату только, если на нем уже установлен элемент ИС:

$$x_{is} \le \sum_{t \in T} y_{st} \qquad \forall i \in I, \quad s \in S. \tag{1.13}$$

Для каждого установленного элемента ИС суммарный требуемый клиентами ресурс не должен превосходить максимально возможную производительность оборудования:

$$\sum_{i \in I} x_{is} b_i \le \sum_{t \in T} prf_t \, y_{st} \qquad \forall s \in S. \tag{1.14}$$

Ограничения на допустимость выделения ресурсов:

$$x_{is} \le \sum_{t \in T} y_{st} \cdot Perm(i,s,t) \quad \forall i \in I, \quad s \in S. \tag{1.15}$$

Здесь *Perm* – функция, значение которой *Perm*(*i*, *s*, *t*) равно единице в случае, если возможно выделение ресурсов элемента типа $t \in T$, установленного на месте $s \in S$, для клиента $i \in I$, иначе – 0. Причины, по которым подобное выделение может быть неосуществимым, зависят от конкретной реализации ИС (например, удаленность клиента от места-кандидата).

Минимизируемая целевая функция имеет вид:

$$\begin{aligned}
F = & \sum_{s \in S} \sum_{t \in T} y_{st} \cdot (1 - \sum_{w \in T} y^0_{sw}) \cdot \left(ElCost_t + PsCost_s \right) + \\
& + \sum_{s \in S} \sum_{t \in T} (1 - y_{st}) \cdot \sum_{w \in T} y^0_{sw} \cdot \left(RemElCost_t + RemPsCost_s \right) + \\
& + \sum_{s \in S} \sum_{t \in T} y_{st} \sum_{z \ne t, z \in T} y^0_{sz} \cdot Upg_{zt} \cdot UpgCost_{zt} + \\
& + \sum_{s \in S} \sum_{t \in T} y_{st} \sum_{z \ne t, z \in T} y^0_{sz} (1 - Upg_{zt}) (RemElCost_z + RemPsCost_s + ElCost_t + PsCost_s) + \\
& + \sum_{i \in I} \sum_{s \in S} x_{is} \cdot (1 - x^0_{is}) \cdot ConnCost_{is} + \\
& + Interf(I,S,T) \cdot k,
\end{aligned} \tag{1.16}$$

где $ElCost_t$ – стоимость элемента типа $t \in T$;

$PSCost_s$ – затраты на установку, зависящие от места-кандидата $s \in S$, например, стоимость долгосрочной аренды места и т.д.;

$RemElCost_t$ – стоимость деинсталляции элемента типа $t \in T$;

$RemPSCost_s$ – затраты на удаление элемента из структуры ИС, зависящие от места-кандидата $s \in S$;

$Upg_{zt} = 1$, если элемент типа $z \in T$ можно модернизировать до типа $t \in T$, иначе – 0; $UpgCost_{zt}$ – стоимость такой модернизации;

$ConnCost_{is}$ – стоимость подключения i-го клиента к элементу на s-м месте (сюда входят, например, затраты на прокладку кабеля).

Таким образом, первая строка формулы (1.16) отвечает за учет затрат по установке новых элементов ИС, вторая строка – за учет затрат по выведению элементов из структуры ИС, третья и четвертая строка – за учет затрат по модернизации элементов ИС, пятая строка – за учет затрат по подключению клиентов к элементам ИС.

Функция $Interf(I, S, T)$ отвечает за учет помех при межмодульном взаимодействии элементов системы. Коэффициент k обеспечивает одновременный учет в целевой функции затрат на модернизацию ИС и уровня помех. k имеет размерность стоимости, тем самым обеспечивая сохранение размерности в формуле (1.16). Стоит отметить, что вид функций *Perm* и *Interf* зависит от конкретной разновидности ЗРЭРИС.

Достоинством приведенной модели является то, что проектирование информационной системы «с нуля» также является её частным случаем ($y_{st}^0 = 0, x_{is}^0 = 0, \forall i \in I, \quad s \in S, \quad t \in T$).

Вышеописанная задача относится к классу задач размещения. Неравенство (1.14) обуславливает принадлежность данной задачи к классу задач размещения с ограничениями на мощности. Ограничение (1.12) обуславливает принадлежность нашей задачи к классу SSCFLP (один клиент обслуживается одним элементом ИС). Функция $Interf(I, S, T)$ в общем случае является нелинейной, а значит, поставленная задача относится к классу задач целочисленного нелинейного программирования.

Принадлежность поставленной задачи к классу задач целочисленного нелинейного программирования обуславливает целесообразность использования эвристических алгоритмов для ее решения. Возможной альтернативой эвристикам являются точные методы нахождения решения, которые реализуют некоторую

ускоренную форму полного перебора. Наиболее распространенными методами данного типа, часто применяющимися к задачам SSCFLP, являются метод Лагранжевых релаксаций и метод ветвей и границ [18,28,34,35]. Однако наиболее известные разновидности данных методов подразумевают переход к двойственной задаче линейного программирования (для получения нижней оценки решения), что невозможно сделать для задачи нелинейной оптимизации [18,34,35]. К тому же данные методы значительно теряют в производительности с ростом числа переменных и ограничений [26,18].

Подход, лежащий в основе эвристических алгоритмов, заключается в отказе от поиска оптимального решения за экспоненциальное время и в попытке нахождения приближенного (достаточно близкого к оптимальному) решения за приемлемое время [18]. Эта особенность эвристик очень важна для ЗРЭРИС, т.к. данная задача имеет явную практическую направленность, и ее решение желательно находить за небольшой промежуток времени. Другой важной особенностью эвристических алгоритмов является тот факт, что для большинства из них не имеет значение, к какому типу относится решаемая проблема – к задачам линейной или нелинейной оптимизации. Более подробное описание и формальное определение эвристик и метаэвристик дано в разделе 2.1.

Решение ЗРЭРИС является одним из этапов процесса принятия решений по планированию и оптимизации информационных систем. Соответственно алгоритмы, позволяющие решать ЗРЭРИС, являются алгоритмами интеллектуальной поддержки принятия решений.

1.4.2 Другая форма записи задачи размещения элементов развивающихся информационных систем

Можно заметить, что формулировка задачи размещения элементов развивающихся информационных систем, приведенная в разделе 1.4.1, является весьма громоздкой. Поэтому встает проблема более лаконичной постановки ЗРЭРИС, удобной для программной реализации. Главной особенностью перехода

будет представление величин x_{is} и y_{st} в виде одномерных массивов X и Y. Отметим, что подобное представление сделает удобным реализацию эволюционного алгоритма, т.к. в эволюционном алгоритме, как правило, хромосомы представляют собой одномерные массивы.

Решение задачи будем представлять в виде пары векторов целых чисел – Y и X. Y – вектор размерности N_{ps}. Элементы массива Y могут принимать целые значения из диапазона $[0; N_{types}]$. Если $Y[s] = 0$, то на s-м месте-кандидате не установлен элемент ИС. Если $Y[s] = w$, то на s-м месте-кандидате установлен элемент ИС w-го типа. X – вектор размерности N_{cl}. Элементы X могут принимать целые значения из диапазона $[1; N_{ps}]$. Если $X[i] = w$, то это значит, что i-й клиент подключен к элементу ИС, установленному на w-м месте-кандидате. Пояснение: здесь и в дальнейшем запись вида $A[j]$ означает обращение к j-му элементу вектора (массива) A; элементы массивов нумеруются, начиная с 1.

Тогда ограничения примут следующий вид. Клиент может быть подключен к месту-кандидату, только если на нем уже установленэлемент ИС:

$$Y[X[i]] \neq 0 \qquad \forall i \in \{1, 2, \ldots N_{cl}\}. \tag{1.17}$$

Пусть b – вектор размерности N_{cl}, элементами которого являются потребности клиентов; prf – вектор максимальных производительностей элементов разного типа (размерности N_{types}). Примечание: будем считать, что $prf[0] = 0$. Ограничение (1.14) примет вид:

$$\sum_{i=1}^{N_{cl}} p_{is} \cdot b[i] \leq prf[Y[s]] \quad \forall s \in \{1, 2 \ldots N_{ps}\},$$
$$\text{где } p_{is} = \begin{cases} 1, & \textit{если } X[i] = s \\ 0 & \textit{иначе} \end{cases}. \tag{1.18}$$

Ограничения (1.15) на допустимость выделения ресурсов примут вид:

$$Y[X[i]] \cdot Perm(i, X[i], Y[X[i]]) \geq 0 \qquad \forall i \in \{1, 2, \ldots N_{cl}\}. \tag{1.19}$$

Отметим, что ограничения (1.11) и (1.12) при форме представления в виде одномерных массивов X и Y выполняются автоматически.

Пусть *ElCost* – вектор стоимостей элементов разного типа; *PsCost* – вектор затрат на установку, зависящих от места-кандидата; *RemElCost* – вектор

стоимостей деинсталляции элементов; *RemPsCost* – вектор затрат на удаление элементов из структуры ИС, зависящих от места-кандидата $s \in S$; *Upg* – двумерный массив (матрица) размерности $N_{types} \times N_{types}$, каждый элемент которого $Upg[z][t]$ равен 1, если элемент ИС типа z можно модернизировать до типа t, иначе – равен 0; *UpgCost* – матрица стоимостей модернизации; *ConnCost* –матрица размерности $N_{cl} \times N_{ps}$, элементы которой $ConnCost[i][s]$ отражают стоимость подключения i-го клиента к элементу на s-м месте.

Тогда целевая функция (1.16) примет вид:

$$
\begin{aligned}
F = & \sum_{s=1}^{N_{ps}} new\_el[s] \cdot \left(ElCost[Y[s]] + PsCost[s] \right) + \\
& + \sum_{s=1}^{N_{ps}} del\_el[s] \cdot \left(RemElCost[Y[s]] + RemPsCost[s] \right) + \\
& + \sum_{s=1}^{N_{ps}} chn\_el[s] \cdot Upg[Y^0[s]][Y[s]] \cdot UpgCost[Y^0[s]][Y[s]] + \\
& + \sum_{s=1}^{N_{ps}} chn\_el[s] \cdot (1 - Upg[Y^0[s]][Y[s]]) \cdot \left(\begin{array}{l} RemElCost[Y^0[s]] + RemPsCost[s] + \\ + ElCost[Y[s]] + PsCost[s] \end{array} \right) + \\
& + \sum_{i=1}^{N_{cl}} new\_conn[i] \cdot ConnCost[i][X[i]] + \\
& + Interf(I,S,T) \cdot k,
\end{aligned}
\tag{1.20}
$$

$$
\begin{aligned}
& \text{где } new\_el[s] = \begin{cases} 1, & если\ Y[s] \neq 0 \quad и \quad Y^0[s] = 0 \\ 0 & иначе, \end{cases} \\
& del\_el[s] = \begin{cases} 1, & если\ Y[s] = 0 \quad и \quad Y^0[s] \neq 0 \\ 0 & иначе, \end{cases} \\
& chn\_el[s] = \begin{cases} 1, & если\ Y[s] \neq 0 \quad и \quad Y^0[s] \neq 0 \quad и \quad Y[s] \neq Y^0[s] \\ 0 & иначе, \end{cases} \\
& new\_conn[i] = \begin{cases} 1, & если\ X[i] \neq X^0[i] \\ 0 & иначе, \end{cases}
\end{aligned}
\tag{1.21}
$$

Требуется отметить, что нет необходимости каждый раз при проверке решения на соответствие ограничениям (1.19) производить расчеты. Достаточно один раз перед началом работы алгоритма выполнить следующие действия. Создать трехмерный массив булевых переменных *perm_constraints* размерности

$N_{cl}\times N_{ps}\times N_{types}$. Рассчитать элементы массива: элемент *perm_constraints*[*i*][*s*][*t*] равен *true*, если ограничения (1.19) выполняются при подключении *i*-го клиента к элементу *t*-го типа, расположенному на *s*-м месте, иначе – *false*. В дальнейшем в процессе решения ЗРЭРИС необходимо вместо проверки ограничений (1.19) просто обратиться к соответствующему элементу массива *perm_constraints*.

1.5 Модель задачи размещения элементов для частного случая информационной системы – беспроводной сети передачи данных

В данном разделе будет рассмотрен частный случай ИС – беспроводная сеть передачи данных. Беспроводная информационная сеть (беспроводная сеть передачи данных, сеть беспроводного доступа, беспроводная вычислительная сеть) – совокупность оборудования, протоколов, специального программного обеспечения, в которой коммуникация ведется без использования кабельной проводки, т.е. основана на беспроводном принципе [144]. Как правило, в виде носителя информации выступают радиоволны СВЧ-диапазона. Согласно ГОСТ Р 55897−2013 [7], сеть радиосвязи это «сеть электросвязи, предназначенная для обеспечения беспроводной связью абонентских станций и представляющая собой совокупность базовых станций (БС), узлов коммутации и линий связи».

Стоит отметить, что беспроводные сети передачи данных сами по себе не являются хранилищами информации (за исключением информации, необходимой для их собственного функционирования). Однако на сегодняшний день одной из главных их функций можно назвать предоставление пользователям доступа в сеть Интернет, которую можно рассматривать как самое крупное в мире хранилище информации.

Основными элементами беспроводных сетей передачи данных являются базовые станции. Базовая станция (Base Transceiver Station/Base Station – BTS/BS) – совокупность оборудования, выполняющего функции приемопередатчика сигнала [142]. Каждая базовая станция осуществляет обслуживание группы оконечных пользовательских устройств (клиентов, абонентских станций) [152].

Согласно ГОСТ Р 55897−2013 [7], базовая станция представляет собой «средство электросвязи, которое размещается стационарно и обеспечивает соединение по радиочастотным каналам множества абонентских станций, находящихся в зоне ее обслуживания, с узлом коммутации сети беспроводной связи».

В качестве вычислительного ресурса, требуемого клиентами, выступают пропускные способности, т.е. трафик.

Таким образом, в случае беспроводных сетей передачи данных мы имеем дело с частным случаем ЗРЭРИС – задачей размещения базовых станций (ЗРБС). Отметим, что в качестве клиентов в ЗРБС выступают так называемые тестовые точки (Test Points, TP) [131]. Каждая TP соответствует группе пользователей, объединенных по какому-либо признаку (например, все компьютеры здания могут составлять одну TP). В данной работе под клиентами в ЗРБС будут подразумеваться именно тестовые точки.

Еще одной особенностью ЗРБС является тот факт, что отсутствуют затраты на подключение клиентов к базовым станциям, т.к. не надо тянуть кабель и т.д.

Модернизация базовых станций происходит следующим образом (на примере модернизации сетей 3-го поколения к стандартам 4-го поколения (4G)). В основе большинства БС стандарта 3G лежат блоки двух типов: BBU (Baseband Unit, блок обработки базовых частот) и RRU (Remote Radio Unit, удаленный радиомодуль).

Модернизация блоков RRU осуществляется, как правило, за счет апгрейда программного обеспечения модуля. Модернизация BBU, если он не поддерживает новые стандарты, осуществляется за счет установки специальных плат расширения в дополнение к существующим. Таким образом, модернизация уже существующей базовой станции оказывается значительно дешевле, чем вывод БС из структуры ИС, ее демонтаж, и установка на ее место новой БС. Именно этим и обусловлено наличие слагаемых, отвечающих за модернизацию элементов системы, в целевой функции (1.16) модели раздела 1.4.1.

Математическая модель задачи размещения базовых станций

Среди русскоязычных публикаций по проблеме размещения базовых станций отметим работы [17] и [18], посвященные сетям WiMAX. Отметим работы [2] и [1] В.М. Вишневского и его коллег, которые являются первыми русскоязычными публикациями по данной проблеме. Подробная классификация работ, посвященных проблеме планирования топологии сетей стандарта UMTS дана в статьях [160,161].

Среди недостатков существующих работ, посвященных рассматриваемой проблеме, отметим следующие:

– решение задачи размещения базовых станций методами, не показывающими высокую скорость расчета в задачах с большим числом ограничений и переменных (метод ветвей и границ, метод Лагранжевых релаксаций и др.) [114];

– отсутствие ограничений, учитывающих уровень затухания сигнала при распространении от базовой станции к клиенту и обратно [1,164];

– в большинстве работ не учитывается уровень межсотовых помех [18,160,175];

– большинство моделей не подразумевают использование нескольких типов базовых станций [160,75,74,76,1,18];

– не рассматривается возможность модернизации уже существующей структуры [160,75,74,76,1,18,175,2,160,161].

Рассмотрение беспроводной сети передачи данных как частного случая информационной системы позволяет нам применить к ЗРБС модель, приведенную в разделе 1.4.1, а также применить к получившейся модели алгоритмы, предложенные в главе 2.

Математическая модель ЗРБС как частного случая ЗРЭРИС имеет следующие специфические особенности.

1) Функция *Perm* из формулы (1.15) принимает вид двух мощностных ограничений. Несмотря на затухание сигнала на пути от базовой станции к ТР,

мощность, доходящая от передатчика к приёмнику, должна превышать минимальную целевую мощность [77]:

$$x_{is} \le \sum_{t \in T} \frac{g_{is} P^t_{bs\_max}}{P^i_{cl\_tar}} y_{st} \qquad \forall i \in I, \quad s \in S. \tag{1.22}$$

где $P^t_{bs\_max}$ – максимальная мощность БС типа $t \in T$; $P^i_{tp\_tar}$ – минимальная мощность, которая должна быть получена клиентом $i \in I$ для нормальной работы (т.н. чувствительность); g_{is} – уровень затухания между клиентом $i \in I$ и местом $s \in S$.

Аналогичное ограничение для режима uplink (несмотря на затухание сигнала на пути от ТР к базовой станции, мощность, доходящая от клиента к БС, должна превышать минимальную целевую мощность [160]):

$$x_{is} \le \sum_{t \in T} \frac{g_{is} P^i_{cl\_max}}{P^t_{bs\_tar}} y_{st} \qquad \forall i \in I, \quad s \in S. \tag{1.23}$$

где $P^i_{tp\_max}$ – максимальная мощность клиента $i \in I$; $P^t_{bs\_tar}$ – чувствительность базовой станции типа $t \in T$.

2) Введем понятие уровеня *SIR* для клиентов системы. *SIR* – отношение уровня сигнала к уровню помех (signal-to-interference ratio) [130]. В общем случае рассчитывается так:

$$SIR_{dB} = 10 \lg \frac{P_{signal}}{P_{interference}}, \tag{1.24}$$

где P_{signal} – мощность сигнала, получаемого клиентом, $P_{interference}$ – мощность помех для данного клиента. Уровень *SIR* характеризует качество сигнала, получаемого от базовой станции, и является важным показателем качества обслуживания.

Тогда функция *Interf* из формулы (1.16) принимает следующий вид:

$$\begin{aligned} &Interf(I,S,T) = \sum_{i \in I} SIR^i_{dB} \cdot k, \\ &SIR^i_{dB} = 10 \lg \frac{\sum_{s \in S} \sum_{t \in T} g_{is} \cdot P^t_{bs\_max} \cdot y_{st} \cdot p_{is}}{N_0 + \sum_{z \in I, z \ne i} \sum_{s \in S} \sum_{t \in T} g_{zs} \cdot P^t_{bs\_max} \cdot y_{st} \cdot p_{zs}}, \end{aligned} \tag{1.25}$$

где SIR^i_{dB} – значение *SIR* для клиента $i \in I$, N_0 – уровень шума (принят равным – 100 дБм) [153]. Мы рассчитываем *SIR* отдельно для каждого клиента. В числителе фигурирует получаемый сигнал от базовой станции, к которой подключен наш клиент, в знаменателе – сигналы от остальных базовых станций, которые создают помехи плюс уровень шума N_0. Таким образом, в задаче оазмещения базовых станций помехи при межмодульном взаимодействии – это межсотовые помехи.

Коэффициент k обеспечивает одновременный учет в целевой функции затрат на создание сети и уровня межсотовых помех для клиентов. k имеет размерность затрат на модернизацию комплекса БС, тем самым обеспечивая сохранение размерности в формуле (1.16), так как *SIR* – безразмерная величина (измеряется в дБ). k должен обеспечивать «штраф» за низкий уровень *SIR* и «награду» за высокий, а значит, должен иметь отрицательное значение, так как мы решаем минимизационную задачу, а значит, хороший (высокий) уровень *SIR* должен уменьшать целевую функцию.

Отметим, что формула (1.25) позволяет вычислить не сам уровень *SIR* для клиента, а величину ему пропорциональную, т.к. не учитывает ряд моментов, специфичных для конкретных стандартов (UMTS, 3GPP, WiMAX) [107].

Величины g_{is} ($i \in I$, $s \in S$) рассчитываются при помощи моделей распространения радиосигнала, которые описывают величину затухания сигнала (потери при распространении). Наиболее широко применяемыми моделями на сегодняшний день являются модель Окумура-Хата [148,115], модель COST231-Хата [89], модель Lee, модель Ikegami, модель Walfisch–Bertoni, модель COST231/Walfisch–Ikegami [154].

Таким образом, математическая модель, предлагаемая в данном разделе, основывается на работах В.М. Вишневского ([1, 2]) и С.Ю. Ермолаева ([18]). Но в отличие от последней, предлагаемая в данной диссертации модель:

– позволяет осуществлять модернизацию уже существующей сети, а не только планирование «с нуля»;

– учитывает уровень *SIR* клиентов;

– учитывает тот факт, что мобильный оператор может использовать несколько типов БС, которые отличаются по своим характеристикам;

– учитывает свойства мест-кандидатов;

– учитывает уровень затухания сигнала при распространении в виде мощностных ограничений (1.22) и (1.23). В [18] используются не мощностные ограничения, а заранее заданный радиус действия БС, что недостаточно корректно, т.к. данная величина зависит от типа местности и других условий распространения сигнала.

1.6 Выводы

1. Проведен системный анализ проблематики интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем. Проведен обзор существующих работ по обозначенной тематике, отмечены основные достоинства и недостатки предлагаемых в них решений

2. Сформулированы цель и задачи диссертационного исследования.

3. Предложена математическая модель задачи размещения элементов развивающихся информационных систем, отличающаяся учетом возможности модернизации элементов системы, свойств мест-кандидатов, помех при межмодульном взаимодействии элементов системы и подразумевающая использование нескольких типов элементов информационной системы.

4. Поставленная задача является задачей целочисленного нелинейного программирования, что позволяет обосновать целесообразность использования эвристических алгоритмов.

2 АЛГОРИТМИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ИНТЕЛЛЕКТУАЛЬНОЙ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ПО ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ ЭЛЕМЕНТОВ РАЗВИВАЮЩИХСЯ ИНФОРМАЦИОННЫХ СИСТЕМ

2.1 Общие сведения о метаэвристических алгоритмах

Согласно [149], задачей комбинаторной оптимизации $P = (S, f)$ называется задача оптимизации, в которой задано конечное множество объектов S и целевая функция $f : S \to \Re^+$, назначающая положительное значение стоимости для каждого из объектов $s \in S$. Необходимо найти объект с минимальным значением стоимости. Объектами, как правило, являются целые числа, подмножества множества элементов, перестановки множества элементов или графовые структуры [72].

Большинство представляющих практический интерес задач комбинаторной оптимизации являются NP-трудными и их точное решение в общем случае требует построения дерева поиска решений экспоненциального размера. Для решения задач комбинаторной оптимизации разработано множество алгоритмов, которые можно разделить на точные и приближенные. Точные алгоритмы гарантированно находят оптимум для любой задачи комбинаторной оптимизации конечного размера за конечное время [149,145]. Однако подобные алгоритмы, как правило, не показывают высокого быстродействия. В связи с этим актуальными вопросами являются разработка и исследование приближенных (в том числе метаэвристических) алгоритмов.

Термин метаэвристика, впервые введенный F. Glover в [111], представляет собой композицию двух греческих слов: «мета» значит «в верхнем уровне, за пределами», «эвристика» происходит от глагола «heuriskein», означающего «найти» [157]. Эвристика – это любая совокупность правил, действий и процедур,

которая находит допустимое решение $\tilde{x} \in X$, где X – множество возможных решений. В идеале $\tilde{x}$ должно совпадать с оптимальным решением x^*. Для большинства эвристик, однако, можно только надеяться, что $f(\tilde{x})$ будет близким к $f(x^*)$ [72].

До того, как термин «метаэвристика» получил широкое распространение, метаэвристики в научной литературе часто назывались современными эвристиками [151]. Самыми известными представителями класса метаэвристических алгоритмов являются следующие: эволюционные алгоритмы (ЭА), метод имитации отжига (ИО), итеративный локальный поиск, алгоритм поиска с запретами (ПЗ), алгоритм мультистарта (МС), алгоритм оптимизации подражанием муравьиной колонии, алгоритм оптимизации подражанием пчелиной колонии, алгоритм оптимизации подражанием иммунной системе, метод роя частиц.

Создатели метаэвристик часто вдохновляются реальными природными процессами. Преломление света, гравитация, приливы и отливы, спиралевидные движения галактик, поведение летучих мышей, птиц, муравьев, пчел, светлячков, волков – вот далеко не полный список явлений, которые легли в основу эффективных метаэвристических алгоритмов [158].

Метаэвристики – это алгоритмы, использующиеся для поиска решения задач, в которых имеется мало вспомогательной информации: часто неизвестно, на что похоже оптимальное решение, неизвестно, каким способом это решение необходимо искать, а метод полного перебора не подходит, т.к. пространство поиска слишком велико. Но если есть некоторое потенциальное решение, его можно проверить и установить, насколько оно хорошо. Т.е. можно узнать качество решения, когда оно доступно [134].

В отличие от точных методов, которые гарантируют оптимальное решение задачи, эвристические методы лишь пытаются дать хорошее, но не обязательно оптимальное решение. Однако время, затрачиваемое точным методом на нахождение оптимального решения сложной проблемы (если, конечно, такой

способ вообще существует), на много порядков больше, чем время, затрачиваемое эвристической для нахождения приближенного решения [136]. Иногда точные методы ищут решение задачи так долго, что становятся неприменимыми для решения реальных прикладных задач. Именно в таких случаях и возникает необходимость прибегнуть к эвристикам.

В дополнение к необходимости находить хорошие решения сложных задач за разумное время, существуют и другие причины для использования эвристик, среди которых можно выделить следующие [136]:

– неизвестно ни одного точного метода решения задачи;

– точный метод существует, но не может быть использован на имеющемся оборудовании;

– эвристический метод является более гибким, чем точный метод, позволяя, например, включение условий, которые трудно смоделировать;

– эвристический метод используется как часть некой глобальной процедуры, которая гарантирует нахождение оптимального решения задачи;

Хороший эвристический алгоритм должен обладать следующими свойствами [136]:

– решение может быть получено за разумное время;

– решение должно быть близким к оптимальному (с высокой вероятностью);

– вероятность получения плохого решения (далекого от оптимального) должна быть низкой.

В чем же отличие эвристических методов от метаэвристических? Проблема с пониманием слова «метаэвристика» заключается в некоторой неудачности самого термина. Дело в том, что, когда мы, например, слышим слово «метаобсуждение», мы понимаем его как «обсуждение об обсуждении», «метаданные» – «данные о данных» и т.д. Поэтому вполне логично было бы подразумевать под «метаэвристикой» «эвристику об эвристике (для эвристики)» [134]. Но данная интерпретация термина была бы неверной:

метаэвристические методы в общем случае не заключаются в том, чтобы при помощи одних эвристических методов каким-то образом улучшить другие эвристические методы (хотя, как будет показано в главе 3, такое возможно) и т.д. Если упрощать, то метаэвристики – это попросту эвристики более высокого порядка [102]. Под «более высоким порядком» имеется в виду следующее.

– Как правило, эвристики – это проблемно-зависимые алгоритмы, направленные на решение конкретной задачи (или типа задач). В свою очередь, метаэвристика – это проблемно-независимый подход, который может применен к решению широкого круга задач.

– Эвристические алгоритмы подразумевают достаточно точное и подробное описание каждого шага, например, способа получения одного решения из другого и т.д. Для метаэвристик же большинство конкретных особенностей задачи представляют собой «черный ящик». Забегая немного вперед, попытаемся объяснить этот момент на примере алгоритма поиска с запретами. Утрированно метаэвристика поиск с запретами может быть сформулирована следующим образом: «в процессе локального поиска надо просматривать не всю окрестность, а игнорировать ту ее часть, которая недавно была использована в качестве текущего решения». Метаэвристика не объясняет, как именно формировать окрестность, как именно сформулировать правило, отсекающее часть окрестности и т.д. Она просто задает основные принципы, которых надо придерживаться. Если же мы откроем «черный ящик», т.е. применим метаэвристику к конкретной задаче, определимся со значениями параметров алгоритма и т.д., то на выходе мы получим эвристику. Например, метаэвристика «поиск с запретами» порождает множество эвристических алгоритмов: «поиск с запретами для решения задачи коммивояжера», «вероятностный поиск с запретами для задачи составления расписания» и т.д.

Таким образом, метаэвристика – проблемно-независимая алгоритмическая основа высокого уровня, предоставляющая набор руководящих принципов и стратегий для разработки эвристических алгоритмов оптимизации [157]. На

сегодняшний день метаэвристики представляют собой наиболее перспективный и быстро развивающийся класс численных методов оптимизации [26].

2.2 Решение задачи размещения элементов развивающихся информационных систем при помощи эволюционного алгоритма

Общие сведения об эволюционных алгоритмах

Эволюционные алгоритмы (ЭА) являются примером так называемых популяционных алгоритмов, для которых характерна работа с набором потенциальных решений, а не с единственным возможным решением. Каждое решение постепенно улучшается и оценивается, а главной особенностью является тот факт, что потенциальное решение влияет на то, как будут улучшены другие решения. Это может произойти из-за того, что хорошие решения либо повлияют на то, какие плохие решения будут отброшены, либо направятся на улучшение плохих решений [110].

Традиционно ЭА включают генетический алгоритм (ГА), эволюционные стратегии (ЭС), эволюционное программирование и генетическое программирование. В данной диссертационной работе мы остановимся на ГА и ЭС, как на наиболее распространенных и эффективных разновидностях ЭА. ЭС были созданы в 60-х годах 20-го века группой немецких ученых под руководством I. Rechenberg [150]. ГА были изначально разработаны H.J. Bremermann в 1958 году [83], а впоследствии были популяризованы американским ученым J.H. Holland [118,81,119], который приспособил их для изучения адаптаций в природе с целью применения данных механизмов и концепций в информатике. Первая его работа, посвященная генетическим алгоритмам, вышла в 1962 году [120], в 1975 году вышла книга J.H. Holland [117], посвященная применению эволюционных концепций в вычислительной математике. Более подробно история эволюционных алгоритмов описана в работе [96].

Заранее оговоримся, что в дальнейшем будем называть предлагаемый в данной работе алгоритм эволюционным. Дело в том, что эволюционные алгоритмы делятся на генетические алгоритмы и эволюционные стратегии [41]. Отличия генетических алгоритмов и эволюционных стратегий заключаются в следующем [41,12].

– Эволюционные стратегии оперируют векторами действительных чисел, а генетические алгоритмы – двоичными векторами. Однако во многих работах ([38] и др.) используется другая терминология: алгоритм, оперирующий действительными числами, называют генетическим алгоритмом с вещественным кодированием. В данной работе эволюционный алгоритм будет работать с векторами целых чисел.

– Разный процесс селекции. При реализации некоторых эволюционных стратегий процедура селекции хромосом и формирования новой популяции предполагает создание промежуточной популяции, состоящей из всех хромосом-родителей и хромосом-потомков. В генетическом же алгоритме популяции родителей и потомков не сливаются [55].

– При реализации эволюционных стратегий сначала производится рекомбинация, а потом селекция. В случае генетических алгоритмов эта последовательность инвертируется.

В некотором смысле можно утверждать, что разделение на генетические алгоритмы и эволюционные стратегии является искусственным. Можно сказать, что существуют эволюционные алгоритмы, параметрами которых являются «способ организации процесса селекции», «последовательность выполнения процедур селекции и рекомбинации» и др. [134]. Комбинируя значения этих параметров, мы можем получать либо эволюционные стратегии, либо генетические алгоритмы.

Таким образом, корректность употребления термина «эволюционный алгоритм» объясняется просто: всякий генетический алгоритм является в то же самое время эволюционным алгоритмом, обратное – неверно; всякая

эволюционная стратегия является в то же самое время эволюционным алгоритмом, обратное – неверно.

Основные понятия

В таблице 2.1 приведены основные термины, применяемые в теории эволюционных алгоритмов.

Таблица 2.1 – Термины, используемые в эволюционных алгоритмах [134]

Термин	**Описание**
особь	потенциальное решение
популяция	конечное множество особей
родитель	особь, отобранная для получения новых особей
потомок	особь, получаемая при помощи размножения из родительских особей
приспособленность	качество решения
функция приспособленности	функция оценки, используемая для определения уровня качества каждой особи в популяции, используется для сравнения альтернативных решений между собой.
ген	часть решения в закодированной форме
хромосома	упорядоченная последовательность генов
генотип (геном)	структура данных особи, используемая в процессе размножения (состоит из одной или нескольких хромосом)
фенотип	совокупность всех признаков и свойств особи, формирующихся в процессе взаимодействия его генотипа и внешней среды [38]
размножение	создание потомков из родительских особей при помощи использования селекции, рекомбинации и мутации
селекция	отбор особей на основе их приспособленности
рекомбинация (кроссовер, скрещивание)	создание потомков с использованием генотипов родительских особей. Аналог природного полового размножения
мутация	случайное изменение в одном или нескольких генах решения. Аналог природного бесполого размножения
поколение	один цикл оценивания, размножения и обновления популяции; либо популяция, создаваемая в каждом таком цикле

Селекция хромосом

Селекция хромосом заключается в выборе особей для последующего их участия в процессе рекомбинации. С точки зрения программной реализации селекция в ЭА заключается в присвоении каждой особи, исходя из ее приспособленности, вероятности ее выбора для последующего скрещивания. Выбор должен производиться согласно принципу естественного отбора, по которому наибольшие шансы на участие в кроссовере имеют особи с наибольшими значениями функции приспособленности. Наиболее распространёнными являются 3 метода селекции [61]: метод рулетки (пропорциональный), ранговый и турнирный. Для ЭС также используется метод панмиксии [38], когда вероятность выбора i-й особи не зависит от значения F_i ее функции приспособленности, а всегда равна $1/N_{pop}$, где N_{pop} – численность (размер) популяции.

Генетические операторы

В эволюционном алгоритме, как правило, применяются два генетических оператора: оператор скрещивания и оператор мутации. Оператор скрещивания отвечает за то, чтобы генотип особи-потомка состоял из частей генотипов его родителей. Самым распространенным типом скрещивания является многоточечное скрещивание, где *n_points* – число точек скрещивания [55]. Оператор мутации выполняется следующим образом – с вероятностью мутации p_{mut} значение гена в хромосоме меняется на альтернативное допустимое.

Простейший эволюционный алгоритм

Ниже приведен псевдокод базового эволюционного алгоритма. Синтаксис псевдокода данного алгоритма и других алгоритмов, приведенных в данной диссертации, описан в приложении 1.

Алгоритм 2.1 Псевдокод базового эволюционного алгоритма

/* Введем обозначения: *Best* – лучшее решение задачи; *fit*(*a*) – значение

функции приспособленности некоторой особи *a*; *Pop* – текущая популяция; *Children* – множество полученных на данной итерации потомков. */

```
1:  Построить начальную популяцию Pop;
2:  Best = ∅;
3:  REPEAT
4:      FOREACH Pi in Pop DO // цикл оценки особей популяции
5:          Рассчитать fit(Pi);
6:          IF (Best == ∅) OR (fit(Pi) > fit(Best)) THEN
7:              Best = Pi;
8:          END IF;
9:      END FOREACH;
10:     Размножение особей популяции Pop (т.е. формирование
        множества потомков Children);
11:     Формирование нового поколения из множеств Pop и Children;
12: UNTIL (условие останова не выполнено) ;
13: RETURN Best;
```

Возможные условия останова для эволюционных алгоритмов:

1) Число поколений N_{gen} превысило некое заданное число.

2) Истекло время, отводимое на работу алгоритма.

Видно, что приведенный выше алгоритм является максимально общим и абстрактным. Отметим три интересных момента.

– Расчет функции приспособленности *fit*. Осуществляется на основе целевой функции особи-решения. В максимизационных задачах, как правило, значение *fit* совпадает с целевой функцией (или пропорционально ей). Для минимизационных задач обычно значение *fit* обратно пропорционально значению целевой функции.

– Размножение. В вышеприведенном алгоритме не специфицировано, как именно это размножение происходит. Просто подразумевается, что по окончании

данного этапа у нас будет некоторое количество потомков, к которым уже применена мутация.

– Формирование нового поколения. Общий алгоритм не описывает то, как именно формируется новое поколение, т.к. это напрямую зависит от того, используем ли мы ГА или ЭС.

Универсальный эволюционный алгоритм

Поставим целью сформулировать такой эволюционный алгоритм, который в зависимости значений управляющих параметров мог бы трансформироваться в некоторые разновидности ЭС или ГА. Введем параметр N_{child} – число потомков, производимое популяцией за одно поколение.

Алгоритм 2.2 Универсальный эволюционный алгоритм

```
/* Введем обозначения: Best – лучшее решение задачи; fit(a) – значение
функции приспособленности некоторой особи a; Pop – текущая популяция;
Children – множество полученных на данной итерации потомков; Tmp –
вспомогательная промежуточная популяция. */
1:  Построить начальную популяцию Pop;
2:  Best = ∅;
3:  REPEAT
4:      FOREACH Ind_i in Pop DO // цикл оценки особей популяции
5:          Рассчитать fit(Ind_i);
6:          IF (Best == ∅) OR (fit(Ind_i) > fit(Best)) THEN
7:              Best = Ind_i;
8:          END IF;
9:      END FOREACH;
10:     Для каждой особи вычислим вероятность выбора для селекции
        P_i (i = 1, 2, …, N_pop) при помощи одного из 4 методов
        (рулеточный, ранговый, турнирный, панмиксия);
```

```
11:     Children = ∅;
12:     FOR j = 1 TO N_child DO    // создание потомков
13:         Выберем из популяции Pop две особи Par_1 и Par_2 на
            основе вероятностей P_i (i = 1, 2, …, N_pop);
14:         Получим потомка Child_j в результате
            n_points-точечного скрещивания особей Par_1 и Par_2;
15:         Children = Children ∪ Child_j;
16:     END FOR;
17:     Tmp = Children;
18:     IF (использ. слияние популяций родителей и потомков) THEN
19:         Tmp = Tmp ∪ Pop;
20:     END IF;
21:     Формирование нового поколения Pop из N_pop лучших особей
        множества Tmp;
22: UNTIL (условие останова не выполнено) ;
23: RETURN Best;
```

Комбинируя параметры вышеприведенного алгоритма легко получить известные нам разновидности ЭА.

1) Классический генетический алгоритм [41] получится при следующих управляющих параметрах: $N_{child} = N_{pop}$, рулеточная селекция, в новую популяцию переходят только особи потомков.

2) Мы получим т.н. $(N_{pop} + N_{child})$ эволюционную стратегию с рекомбинацией [102] при следующем наборе параметров: селекция по типу панмиксии, создается промежуточная популяция из всех родителей и потомков.

3) Мы получим т.н. (N_{pop}, N_{child}) эволюционную стратегию с рекомбинацией [102] при следующем наборе параметров: селекция по типу панмиксии, в новую популяцию переходят только особи потомков.

Эволюционный алгоритм для решения ЗРЭРИС

Каждая особь-решение состоит из двух векторов-хромосом Y и X, которые были описаны в разделе 1.4.2. Можно заметить, что для каждого решения поставленной в главе 1 задачи вектор X полностью определяет, на каких местах в векторе Y будут стоять не нули. Вектор Y определяет только то, какой именно тип элемента установлен на конкретном месте. Так как хромосомы X и Y зависимы друг от друга, то мы не можем скрещивать их по отдельности (т.к. будет порождаться много недопустимых решений). Поэтому мы будем скрещивать только векторы X_1 и X_2 родительских особей [156,51].

Оператор скрещивания выполняется следующим образом: из родительской популяции случайным образом выбирается пара особей. Далее для каждой пары отобранных таким образом особей разыгрывается позиция гена в хромосоме X, определяющая так называемую точку скрещивания. Эта точка представляет собой целое число l_c из интервала $[1, L-1]$, где L – длина хромосомы X (в нашем случае $L = N_{cl}$). В результате скрещивания двух родительских хромосом получается потомок, хромосома X_{child} которого на позициях от 1 до l_c состоит из генов первого родителя, а на позициях от (l_c+1) до L – из генов второго родителя. Модификацией одноточечного кроссовера является *n_points*-точечный (т.н. многоточечный) кроссовер. Он аналогичен одноточечному, однако в нем скрещивание проходит уже по *n_points* точкам.

К получившемуся вектору X_{child} потомка будет применяться операция мутации. Суть операции мутации заключается в том, что с некоторой небольшой вероятностью p_{mut} один из элементов вектора X_{child} поменяет свое значение на случайное целое число из диапазона $[1; N_{ps}]$.

После получения вектора X_{child} особи-потомка, мы должны сформировать его хромосому Y_{child}. Хромосома X_{child} однозначно определяет, какие элементы вектора Y_{child} являются ненулевыми [51]. Например, у нас есть 5 мест-кандидатов, 4 клиента и 3 типа элементов ИС (причем, чем меньше порядковый номер типа элементов, тем дешевле такие элементы). Пусть $X_1 = [1\ \ 5\ \ 2\ \ 3]$ и $X_2 = [2\ \ 5\ \ 5\ \ 3]$.

Пусть в результате скрещивания для потомка мы получили $X_{child} = [1\ \ 5\ \ 5\ \ 3]$. Это значит, что хромосома Y_{child} потомка будет иметь вид $Y_{child} = [V\ \ 0\ \ V\ \ 0\ \ V]$, где V – целое число из диапазона $[1; N_{types}]$.

Ненулевые элементы вектора Y_{child} рекомендуется выбирать так:

1) если $Y_1[i] = w$, а $Y_2[i] = 0$, то $Y_{child}[i] = w$;

2) если $Y_2[i] = w$, а $Y_1[i] = 0$, то $Y_{child}[i] = w$;

3) если $Y_1[i] = Y_2[i] = w$, то $Y_{child}[i] = w$;

4) если $Y_1[i] = w$, а $Y_2[i] = z$ $(w \neq z)$, то для выбора типа элемента ИС, который будет установлен у потомка на i-м месте необходимо брать либо тип w, либо тип z случайно с равной вероятностью 0.5 [55].

Псевдокод процедуры скрещивания для ЗРЭРИС приведен в Приложении 2 (алгоритм П2.1).

2.3 Алгоритмы локального поиска для решения задачи размещения элементов развивающихся информационных систем

Общие сведения об алгоритмах локального поиска

Локальный поиск (Local Search) – целый класс алгоритмов, для представителей которого характерны следующие особенности:

– алгоритмы локального поиска являются итеративными;

– на каждом шаге алгоритма для текущего решения рассматривается некоторое множество соседних решений – так называемая окрестность текущего решения (или ее часть);

– в каждый момент времени у алгоритма локального поиска есть только одно текущее решение (в отличие, например, от ЭА, у которого есть N_{pop} текущих решений).

В первую очередь рассмотрим одного из простейших представителей класса алгоритмов локального поиска – алгоритм локального спуска. На каждом шаге для текущего решения полностью просматривается окрестность текущего решения. В качестве следующего из нее выбирается решение, доставляющее

минимум целевой функции (если речь идет о задаче минимизации). Процесс продолжается до тех пор, пока в окрестности имеются решения лучшие, чем текущее [65].

Ниже представлен простейший алгоритм локального спуска для задачи минимизации [56]. Предполагается, что вектор x – решение некоторой оптимизационной задачи. Множество всех возможных векторов обозначим X. Пусть требуется минимизировать некоторую функцию $f(x)$ на множестве X.

Алгоритм 2.3 Алгоритм локального спуска

```
    /* Введем обозначения: x0 – начальное решение задачи. */
1:  x = x0;
2:  Построить окрестность соседних решений N(x);
3:  Найти такое решение z ∈ N(x), что f(z) ≤ f(y), для всех
    y ∈ N(x);
4:  IF (f(z) < f(x)) THEN
5:      x = z;
6:      переход к команде 2;
7:  END IF;
8:  RETURN x;
```

Получение окрестности решения ЗРЭРИС для алгоритмов локального поиска

Нам необходимо отсортировать типы элементов ИС по возрастанию цены. В дальнейшем мы будем исходить из того, что i-й тип дороже $(i - 1)$-го и дешевле $(i + 1)$-го.

Понятие «окрестность» является самым интересным в метаэвристиках, основанных на локальном поиске. Оно плохо формализовано и для каждой конкретной задачи оптимизации обладает своей спецификой. В случае ЗРЭРИС у нас имеется решение, представленное в виде ряда элементов с подключенными к ним клиентами. Мы должны определиться, какие решения являются наиболее

близкими к нашему. В данной работе предлагается способ формирования окрестности решения посредством осуществления небольших изменений в текущем решении *Sol*. Новое решение из окрестности текущего решения можно получить одним из шести способов (операций, применяемых к решению), которые порождают 6 типов окрестностей [45].

1. *Смена типа одного элемента на более дешевый*. Максимальное число решений в такой окрестности $N_{cheap\_el}(Sol)$ равно N_{ps}.

2. *Смена типа одного элемента на более дорогой*. Максимальное число решений в такой окрестности $N_{exp\_el}(Sol)$ равно N_{ps}.

3. *Переподключение одного клиента* (т.е. подключение к другому элементу системы). Данная операция возможна для каждого клиента. Если мы ищем новый элемент для *i*-го клиента, то мы должны последовательно [проверяя на соответствие ограничениям (1.14) и (1.15)] пробовать подключить его к одному из мест-кандидатов (кроме его текущего места), начиная с самого ближнего к клиенту *i*. Очевидно, что место *w*, к которому мы хотим подключить клиента, должно иметь активный элемент. Максимально возможное число решений в такой окрестности $N_{reconn\_cl}(Sol)$ равно N_{cl}.

4. *Удаление одного элемента*. Данная операция возможна для каждого активного места-кандидата. Для каждого из клиентов удаляемого элемента *s* мы запускаем операцию 'Переподключение одного клиента'. Если все клиенты *s*-го элемента системы можно подключить к другим элементам, значит удаление *s*-го элемента возможно. Максимальное число решений в такой окрестности $N_{remove\_el}(Sol)$ равно N_{ps}.

5. *Добавление одного элемента*. Данная операция возможна для каждого пустого места. Мы устанавливаем на место *s* новый элемент и пробуем подключить к нему самый близкий клиент (пусть его номер *i*). Если такое подключение возможно [для *s*-го места выполняются ограничения (1.14) и (1.15)], то новое решение *Sol* является допустимым. Клиент с номером *i*, естественно, необходимо отключить от своего старого элемента. Максимальное число решений

в такой окрестности $N_{add\_el}(Sol)$ равно $(N_{ps} - 1)$ (когда только одно место-кандидат имеет установленный элемент).

6. *Перемещение одного элемента*. Данная операция возможна для каждого активного места-кандидата. Пусть мы перемещаем элемент с места *s*. Тогда мы должны последовательно пробовать поместить ее на пустые места-кандидаты, начиная с самого ближнего к месту *s*, с учетом того, что на новом месте *w* должны выполняться ограничения (1.15). Максимальное число решений в такой окрестности $N_{move\_el}(Sol)$ равно $(N_{ps} - 1)$.

Отметим, что все вышеприведенные окрестности являются окрестностями полиномиальной мощности. Полная окрестность некоторого решения *Sol* получается как объединение вышеприведенных шести окрестностей:

$$N(Sol) = N_{cheap\_el}(Sol) \cup N_{exp\_el}(Sol) \cup N_{reconn\_cl}(Sol) \cup \\ \cup N_{remove\_el}(Sol) \cup N_{add\_el}(Sol) \cup N_{move\_el}(Sol). \quad (2.1)$$

Алгоритм локального спуска для решения ЗРЭРИС

Отметим, что применение операций «Удаление одного элемента» и «Смена типа одного элемента на более дешевый» к текущему решению гарантировано уменьшает значение целевой функции (1.16). Применение операций «Смена типа одного элемента на более дорогой» и «Добавление одного элемента» гарантировано увеличивает значение целевой функции (1.16). А применение операций «Переподключение одного клиента» и «Перемещение одного элемента» может как увеличить значение целевой функции, так и уменьшить его.

Таким образом, для некоторого решения *Sol*, решения с лучшей целевой функцией можно получить из окрестности $N_{better}(Sol)$, формируемой при помощи применения одной из операций «Смена типа одного элемента на более дорогой», «Добавление одного элемента», «Переподключение одного клиента», «Перемещение одного элемента» к *Sol*. Отметим, что окрестность $N_{better}(Sol)$ также содержит решения хуже, чем *Sol*:

$$N_{better}(Sol) = N_{cheap\_el}(Sol) \cup N_{reconn\_cl}(Sol) \cup \\ \cup N_{remove\_el}(Sol) \cup N_{move\_el}(Sol). \quad (2.2)$$

Для алгоритма локального спуска, применяемого для решения ЗРЭРИС, будет использоваться окрестность $N_{better}(Sol)$, а не полная окрестность $N(Sol)$. Использование окрестности $N(Sol)$ уместно в алгоритмах имитации отжига и поиска с запретами, т.к. в них допускается переход к решению, которое хуже текущего. Ниже представлена общая схема алгоритма локального спуска для задачи размещения элементов развивающихся систем.

Алгоритм 2.4 Псевдокод алгоритма локального спуска для ЗРЭРИС

```
1:  Построить first – начальное решение;
2:  Sol = first;
3:  REPEAT
4:      Построить окрестность соседних решений Nbetter(Sol);
5:      Найти такое решение z ∈ Nbetter(Sol), что F(z) ≤ F(y), для
        всех y ∈ Nbetter(Sol);
6:      IF (F(z) < F(Sol)) THEN
7:          Sol = z;
8:      ELSE
9:          BREAK;
10:     END IF;
11: UNTIL (TRUE);
12: RETURN Sol;
```

2.4 Решение задачи размещения элементов развивающихся информационных систем при помощи алгоритма имитации отжига

Общие сведения об алгоритме имитации отжига

Представленный выше алгоритм локального спуска относится к алгоритмам локальной оптимизации. Он остановится в первом же найденном локальном

оптимуме. Есть некоторая вероятность, что таким образом будет найден глобальный оптимум, т.к. всякий глобальный оптимум является в то же самое время и локальным. Но в общем случае глобальный оптимум найден не будет.

Представленные в главе 2 алгоритмы, относящиеся к классу алгоритмов локального поиска (поиск с запретами, имитация отжига, мультистарт) являются алгоритмами глобальной оптимизации, т.к. каждый из них в случае достаточно продолжительной работы рано или поздно найдет глобальный оптимум [134]. В алгоритмах поиска с запретами и имитации отжига (ИО) это осуществляется за счет ненулевой вероятности перехода к худшему решению на одной из итераций, в алгоритме мультистарта – за счет перезапуска поиска из нового начального решения.

Алгоритм имитации отжига (Simulated Annealing, SA) берет свое начало от алгоритма Метрополиса [138], созданного в 1953 году бывшими участниками проекта «Манхэттен». Алгоритм Метрополиса позволял осуществлять симуляцию установления равновесия в системе с множеством степеней свободы при заданной температуре. Он основывался на имитации физического процесса, происходящего при кристаллизации вещества (в т.ч. и при отжиге металлов). Предполагается, что атомы уже выстроились в кристаллическую решетку, но еще допустимы переходы отдельных атомов из одной ячейки в другую. Процесс протекает при постепенно понижающейся температуре [44]. Переход атома из одной ячейки в другую происходит с некоторой вероятностью, причем вероятность уменьшается с понижением температуры. Устойчивая кристаллическая решетка соответствует минимуму энергии атомов, поэтому атом либо переходит в состояние с меньшим уровнем энергии, либо остается на месте [43].

В начале 80-х у S. Kirkpatrick [127] появилась идея использовать метод Метрополиса не только для моделирования физических систем, но и для решения задач комбинаторной оптимизации. Примерно в одно время с S. Kirkpatrick, независимо от него похожая идея пришла в голову V. Černý [86].

Автору диссертации неизвестны научные работы, посвященные применению алгоритма имитации отжига для решения задачи размещения элементов развивающихся информационных систем

Простейший алгоритм имитации отжига

На каждом шаге алгоритма имитации отжига в окрестности текущего решения x выбирается некоторое решение w. Если целевая функция $f(w)$ решения w будет лучше (в случае минимизации – меньше), чем $f(x)$, то w заменяет x в качестве текущего решения. Т.е. на каждом шаге алгоритма просматривается не полная окрестность, как в алгоритме локального спуска, а одно случайное решение из нее.

Алгоритм ИО имеет одну особенность, которая помогает ему избегать застревания в локальных оптимумах: есть некоторая вероятность перехода к решению w, даже если $f(w)$ хуже, чем f(x). Эта вероятность для задачи минимизации выражается формулой [127]

$$P(t) = e^{\frac{f(x)-f(w)}{t}}, \tag{2.3}$$

где параметр t по аналогии с физическим процессом называется температурой. Очевидно, что чем выше значение температуры, тем больше вероятность перехода к худшему решению. В процессе работы алгоритма параметр t постоянно уменьшается. В данной работе рассматривается экспоненциальный отжиг, а значит температура на $(k + 1)$-м шаге может быть вычислена при помощи температуры на k-м шаге по формуле:

$$t_{k+1} = t_k \cdot \alpha, \tag{2.4}$$

где параметр $\alpha \in (0; 1)$ – коэффициент охлаждения. Алгоритм прекращает свою работу, когда значение температуры становится близким к нулю. В данной диссертации было принято решение остановиться на экспоненциальном отжиге по двум причинам. Во-первых, среди методов уменьшения температуры этот способ является самым распространенным и эффективным [127, 90, 170, 102]. Во-вторых, в отличие, например, от имитации отжига с фиксированным числом

итераций, экспоненциальный отжиг допускает остановку алгоритма в случае, когда истекло время, отводимое на поиск.

Ниже представлена общая схема алгоритма имитации отжига (для задачи минимизации).

Алгоритм 2.5 Алгоритм имитации отжига

/* Введем обозначения: x_0 – начальное решение задачи; x_{curr} – текущее решение задачи; x_{pot} – потенциальное решение задачи; x_{best} – лучшее решение задачи. */

1: $x_{curr} = x_0$;

2: $x_{best} = x_0$;

3: Присвоить начальное значение температуре t;

4: x_{pot} – случайное решение из окрестности $N(x_{curr})$;

5: IF ($f(x_{pot}) < f(x_{curr})$ OR Random < exp(($f(x_{curr}) - f(x_{pot})$)/t)) THEN

6: $x_{curr} = x_{pot}$;

7: END IF;

8: IF ($f(x_{curr}) < f(x_{best})$) THEN

9: $x_{best} = x_{curr}$;

10: END IF;

11: Уменьшить t;

12: IF (условие останова не выполнено AND $t > 0$) THEN

13: переход к команде 4;

14: END IF;

15: RETURN x_{best};

Возможные условия останова для алгоритмов локального поиска:

1) Истекло время, отводимое на работу алгоритма.

2) Число итераций превысило некое заданное число.

3) Текущее решение является оптимальным.

4) Окрестность соседних решений – пустое множество.

5) Окрестность соседних решений не содержит ни одного решения, улучшающего целевую функцию.

6) Целевая функция не улучшалась некоторое заданное число итераций подряд.

Алгоритм имитации отжига для решения ЗРЭРИС

Особенностью предлагаемого в данной диссертации алгоритма имитации отжига является использование полной окрестности $N(Sol)$, состоящей из решений, получаемых, при помощи применения одной из шести операций, описанных в разделе 2.3, к решению *Sol*. Таким образом, за счет операций «Смена типа одного элемента на более дорогой», «Добавление одного элемента», «Перемещение одного элемента» и «Переподключение одного клиента» в ИО обеспечивается ненулевая вероятность перехода к худшему решению на одной из шагов алгоритма [44].

Отдельно поговорим о выборе начальной температуры t_0, т.к. этот параметр очень важен для корректной работы алгоритма. Авторы алгоритма ИО [127] дают только общий совет, заключающийся в том, что значение t_0 надо выбирать очень большим. Однако в работе [90] была выведена формула для расчета t_0. Пусть p_0 – целевая вероятность принятия худшего решения на ранних шагах алгоритма ИО, Δf_{max} – максимально возможная разница между значениями целевой функции двух соседних решений. Тогда формула для t_0 имеет вид:

$$t_0 = -\frac{\Delta f_{\max}}{\mathrm{Ln}(p_0)}. \qquad (2.5)$$

Очевидно, что заранее вычислить Δf_{max} очень сложно, поэтому мы должны использовать в расчетах приблизительную оценку этой величины. В задаче размещения элементов ИС параметр Δf_{max} должен быть примерно равен стоимости самого дорогого из доступных для установки элементов [44].

Отметим, что при каждом значении температуры можно делать несколько итераций алгоритма (обозначим эту величину как *N_iter*).

Ниже представлена общая схема алгоритма имитации отжига для задачи размещения элементов развивающихся систем [44, 156].

Алгоритм 2.6 Псевдокод алгоритма имитации отжига для ЗРЭРИС

```
/* Введем обозначения: curr_n_iter – текущее число итераций при данном
   значении температуры; t_min – температура, при которой отжиг
   прекращается; Curr – текущее решение; Sol – потенциальное решение; Best –
   лучшее решение задачи; p – вероятность перехода к худшему решению.  */
1:  Построить Curr – начальное решение ЗРЭРИС;
2:  Best = Curr;
3:  Найти t0 по формуле (2.5);
4:  t = t0;
5:  curr_n_iter = 0;
6:  REPEAT
7:      curr_n_iter++;
8:      Sol = Curr;
9:      Применить к Sol одну из 6 операций по изменению решения
        (выбор операции осуществляется случайно);
10:     IF (Sol – недопустимое решение) THEN
11:         CONTINUE;
12:     END IF;
13:     P = exp((F(Curr) – F(Sol)) / t);
14:     IF ((F(Sol) < F(Curr)) OR (Random < P)) THEN
15:         Curr = Sol;
16:         IF (F(Curr) < F(Best)) THEN
17:             Best = Curr;
18:         END IF;
19:     END IF;
20:     IF (curr_n_iter = N_iter) THEN
21:         t = t × α;
22:         curr_n_iter = 0;
23:     END IF;
24: UNTIL ((условие останова не выполнено) AND (t > t_min));
25: RETURN Best;
```

2.5 Решение задачи размещения элементов развивающихся информационных систем при помощи алгоритма поиска с запретами

Общие сведения об алгоритме поиска с запретами

Поиск с запретами (Tabu Search, TS) является одним из наиболее эффективных метаэвристических алгоритмов. Основателем метода является F. Glover, который предложил его в 80-е годы [112,113]. Отличительной чертой поиска с запретами (ПЗ) является процесс введения и снятия некоторых искусственных ограничений задачи в процессе поиска решения [65].

В алгоритме поиска с запретами для обеспечения возможности перехода из одного локального оптимума в другой вводится так называемый список запретов (Tabu List, TL). Этот список хранит некоторое количество предыдущих решений, и при выборе нового решения запрещается выбирать из окрестности решения, содержащиеся в списке запретов [134]. Данный прием позволяет избегать застревания в локальном оптимуме, расширяет пространство поиска, позволяя алгоритму поиска с запретами находить лучшие решения, чем метод локального спуска.

Список запретов *TL* строится по предыстории поиска, т.е. по нескольким последним итерациям, и запрещает часть полной окрестности $N(x)$ текущего решения [56]. Таким образом, поиск с запретами представляет собой разновидность локального спуска, в процессе которого осуществляется поиск не в полной окрестности $N(x)$, а в окрестности $N_{TL}(x)$, получаемой при помощи удаления из $N(x)$ решений, находящихся в списке запретов.

На каждой итерации поиска с запретами список запретов обновляется. Обновление списка запретов состоит из двух этапов: удаление самого «старого» элемента списка (если текущая длина списка равна максимальной длине l) и добавление в список нового элемента.

Ниже приведен общий алгоритм поиска с запретами (для задачи минимизации).

Алгоритм 2.7 Алгоритм поиска с запретами

/* Введем обозначения: x_0 – начальное решение задачи; x_{curr} – текущее решение задачи; x_{best} – лучшее решение задачи. */

1: $x_{curr} = x_0$;

2: $x_{best} = x_0$;

3: $TL = \varnothing$;

4: Добавить решение x_0 в TL;

5: Построить окрестность соседних решений $N_{TL}(x_{curr})$;

6: Найти такое решение $z \in N_{TL}(x_{curr})$, что $f(z) \leq f(y)$, для всех $y \in N_{TL}(x_{curr})$;

7: $x_{curr} = z$;

8: Обновить список TL;

9: IF ($f(x_{curr}) < f(x_{best})$) THEN

10: $x_{best} = x_{curr}$;

11: END IF;

12: IF (условие останова не выполнено) THEN

13: переход к команде 5;

14: END IF;

15: RETURN x_{best};

Вероятностный поиск с запретами

Вероятностный поиск с запретами впервые был предложен в работе [106]. Рассмотрим рандомизированную окрестность $N_p(x) \subseteq N(x)$, где каждый элемент окрестности $N(x)$ включается в множество $N_p(x)$ с вероятностью $0 \leq p \leq 1$ (p – параметр рандомизации окрестности) независимо от других элементов [40]. С ненулевой вероятностью множество $N_p(x)$ может совпадать с $N(x)$, может оказаться пустым или содержать ровно один элемент. Алгоритм поиска с запретами, представленный в данном разделе, осуществляет вероятностный

локальный поиск по рандомизированной окрестности (поэтому он называется вероятностный ПЗ), совершая шаги как улучшающие целевую функцию, так и ухудшающие ее, что позволяет алгоритму перемещаться от одного локального оптимума к другому в целях найти среди них лучшее решение [46]. По аналогии с классическим ПЗ, в вероятностном ПЗ список запретов накладывает некоторые ограничения на множество потенциальных решений, и поэтому поиск осуществляется не по окрестности $N_p(x)$, а по окрестности $N_{p-TL}(x)$, получаемой при помощи удаления из $N_p(x)$ решений, находящихся в списке запретов.

В работах [106,26,40,5,46] показано, что вероятностный поиск с запретами в среднем эффективней поиска с запретами, при котором просматривается вся окрестность. Очевидно, что при $p = 1$ окрестность $N_p(x)$ совпадает с $N(x)$.

Псевдокод алгоритма вероятностного поиска с запретами аналогичен псевдокоду классического ПЗ с одним изменением: $N_{TL}(x)$ необходимо заменить на $N_{p-TL}(x)$.

Алгоритм вероятностного поиска с запретами для решения ЗРЭРИС

Первой особенностью предлагаемого в данной диссертации алгоритма поиска с запретами является использование рандомизированной окрестности $N_{p\text{-}TL}(Sol)$, получаемой из полной окрестности $N(Sol)$. Та, в свою очередь, состоит из решений, получаемых, при помощи применения одной из шести операций, описанных в разделе 2.3, к решению *Sol*.

Другой особенностью является следующий момент. Как уже было сказано, в классическом алгоритме поиска с запретами в список запретов непосредственно добавляются те решения, которые были признаны лучшими на соответствующей итерации и к которым был осуществлен переход как к новому текущему решению. Однако специфика нашей задачи такова, что для нее характерно очень большое пространство поиска, в котором легко застрять в локальной окрестности, принадлежащей одному пику, даже если использовать очень большой список запретов. Возможных решений может быть слишком много.

Поэтому предлагается в список запретов добавлять не решения, а операции, которые нельзя будет осуществлять в течение *l* следующих итераций. То есть каждое изменение, примененное к решению, порождает запрет на операцию, которая может вернуть нас в это самое решение [46]. Ниже приведены пары «Изменение» – «Запрет» (пояснение: фразой типа «*s*-й элемент» для краткости обозначается элемент, расположенный на *s*-м месте):

1) «Смена типа *s*-го элемента на более дешевый» – «смена типа *s*-го элемента на более дорогой».

2) «Смена типа *s*-го элемента на более дорогой» – «смена типа *s*-го элемента на более дешевый».

3) «Переподключение *i*-го клиента» – «переподключение *i*-го клиента».

4) «Удаление *s*-го элемента» – «добавление *s*-го элемента».

5) «Добавление *s*-го элемента» – «удаление *s*-го элемента».

6) «Перемещение *s*-го элемента на *w*-е место» – «перемещение *w*-го элемента».

Еще одной особенностью предлагаемого в данной диссертации алгоритма поиска с запретами является использование диверсификации в процессе поиска. Данный прием, позволяющий улучшить функционирование алгоритма ПЗ, подразумевает перезапуск поиска с запретами из нового начального решения, если не удаётся улучшить целевую функцию определенное число итераций подряд. Для удобства обозначим как *use_divers* булеву переменную, показывающую, используем ли мы стратегию диверсификации в процессе вероятностного ПЗ. Ниже представлена общая схема вероятностного поиска с запретами для задачи размещения элементов развивающихся систем.

Алгоритм 2.8 Псевдокод алгоритма вероятностного поиска с запретами для решения задачи размещения элементов развивающихся систем

/* Введем обозначения: *Best_neigh* – лучшее решение в текущей окрестности; *cost_best_neigh* – целевая функция решения *Best_neigh*; *Best* – лучшее решение задачи; *cost_best* – целевая функция *Best*. */

```
1:  Best = ∅;
2:  cost_best = ∞;
3:  REPEAT
4:      TL = ∅;
5:      Построить Curr – начальное решение ЗРЭРИС;
6:      REPEAT
7:          Построить окрестность N_TL(Curr);
8:          Best_neigh = ∅;
9:          cost_best_neigh = ∞;
10:         FOREACH Sol in N_TL(Curr) DO
11:             IF (Random > p) THEN
12:                 CONTINUE;
13:             END IF;
14:             IF (Sol – допустимое решение) THEN
15:                 IF (F(Sol) < cost_best_neigh) THEN
16:                     Best_neigh = Sol;
17:                     cost_best_neigh = F(Sol);
18:                 END IF;
19:             END IF;
20:         END FOREACH;
21:         IF (cost_best_neigh != ∞) THEN
22:             Curr = Best_neigh;
23:         END IF;
24:         Обновить TL;
25:         IF (F(Curr) < cost_best) THEN
26:             Best = Curr;
27:         END IF;
```

```
28:         IF (условие останова выполнено) THEN
29:             RETURN Best;
30:         END IF;
31:         IF (use_divers = FALSE) THEN
32:             CONTINUE;
33:         END IF;
34:         IF (решение Curr не улучшалось некоторое заранее
            определенное число итераций подряд) THEN
35:             BREAK;
36:         END IF;
37:     UNTIL (TRUE);
38: UNTIL (условие останова не выполнено);
39: RETURN Best;
```

В приложении 2 (рисунок П2.1) приведена блок-схема вероятностного поиска с запретами для задачи размещения элементов развивающихся систем.

2.6 Решение задачи размещения элементов развивающихся информационных систем при помощи алгоритма мультистарта

Общие сведения об алгоритме мультистарта

Как уже было сказано, глобальный оптимум является в то же самое время одним из локальных оптимумов. Соответственно, если просмотреть все локальные оптимумы и выбрать из них лучший, то это даст нам окончательное решение задачи. Алгоритм мультистарта (Multi-Start, MS) нацелен на то, чтобы обойти как можно большее число локальных оптимумов [110]. Суть метода проста: алгоритм локального спуска (вернее, его модификация) запускается несколько раз. Лучшее решение, полученное при каждом запуске, сохраняется в памяти. По окончании времени поиска лучший из локальных оптимумов возвращается в качестве решения задачи.

Алгоритм мультистарта (MC) даст хорошие результаты при выполнении двух условий [110]:

– каждый запуск алгоритма (будем называть его старт) должен стартовать из нового начального решения;

– сама процедура локального поиска должна быть выполнена так, чтобы каждый новый старт по возможности приводил в новый локальный оптимум. Это необходимо для расширения пространства поиска.

Алгоритм мультистарта показывает результаты, сопоставимые с другими метаэвристиками, и поэтому часто используется для решения NP-трудных задач [137,82,139]. Ниже приведен псевдокод базового алгоритма мультистарта для задачи минимизации [60].

Алгоритм 2.9 Алгоритм мультистарта

```
1:  i = 0;
2:  REPEAT
3:      i++;
4:      Построить x_i – начальное решение для i-го старта;
5:      REPEAT
6:          Построить окрестность соседних решений N(x_i);
7:          Найти такое решение z ∈ N(x_i), что f(z) ≤ f(y), для
            всех y ∈ N(x_i);
8:          IF (f(z) < f(x_i)) THEN
9:              x_i = z;
10:         ELSE
11:             BREAK;
12:         END IF;
13:     UNTIL (TRUE);
14: UNTIL (условие останова не выполнено);
15: RETURN лучшее из решений x_i;
```

Алгоритм мультистарта для решения ЗРЭРИС

Введем понятие жадного алгоритма. В жадном алгоритме делается выбор, который является самым лучшим в данный момент, т.е. производится локально оптимальный выбор в надежде, что он приведет к оптимальному решению глобальной задачи [6]. В предлагаемом в данной диссертации алгоритме мультистарта используется жадная эвристика «первое улучшение» [110]. Суть ее заключается в том, что при поиске мы рассматриваем одно случайное соседнее решение z из окрестности текущего решения Sol (а не всю окрестность). Если $f(z)$ лучше, чем $f(Sol)$, то текущим решением становится z. Подобный переход к первому же лучшему решению расширяет пространство поиска, что в сочетании со стартами из разных начальных решений позволит попасть в большее количество разных локальных оптимумов.

Новые потенциальные решения мы будем генерировать не из окрестностей $N(Sol)$ или $N_p(Sol)$, а из окрестности $N_{all\_better}(Sol)$. $N_{all\_better}(Sol)$ – окрестность, получаемая при помощи применения операций «Удаление одного элемента» и «Смена типа одного элемента на более дешевый» к Sol. Таким образом, $N_{all\_better}(Sol)$, в отличие от $N(Sol)$, содержит только решения с целевой функцией заведомо лучше, чем у Sol.

$$N_{all\_better}(Sol) = N_{cheap\_el}(Sol) \cup N_{remove\_el}(Sol). \qquad (2.6)$$

Так как в предлагаемом нами алгоритме просматривается не вся окрестность, а лишь отдельные случайные ее элементы, необходимо пересмотреть и критерий окончания каждого из стартов. Предлагается заканчивать отдельный старт в случае, когда число итераций подряд без улучшения целевой функции превысит некое заранее заданное число *N_iter_max* [134, 110]. Т.е. если мы предприняли *N_iter_max* неудачных попыток улучшить некоторое решение Sol, то текущий старт считается законченным, результатом его выполнения является решение Sol.

Другой особенностью предлагаемого в данной работе алгоритма мультистарта является концепция интенсификации поиска в т.н. перспективных

областях [133,110]. Суть интенсификации заключается в том, что если по окончании старта (пусть его номер i) его результат $Curr_i$ будет достаточно близок к текущему лучшему решению задачи *Best*, необходимо из точки $Curr_i$ запустить локальный поиск с более тщательным просмотром окрестности (по сравнению с генерацией одного случайного решения). В работе [133] предлагается запускать дополнительный локальный спуск для решений, удовлетворяющих условию

$$F(Curr_i) < \theta \cdot F(Best). \qquad (2.7)$$

Величину θ назовем параметром интенсификации поиска. Авторы работы [133] утверждают, что значение $\theta = 1.25$ обеспечивает наилучший компромисс между скоростью получения решения и его качеством.

Ниже представлена общая схема алгоритма мультистарта для задачи размещения элементов развивающихся информационных систем.

Алгоритм 2.10 Псевдокод алгоритма мультистарта для ЗРЭРИС

```
/* Введем обозначения: N_iter_max – максимально возможное число
   итераций алгоритма подряд без улучшения целевой функции; n_iter –
   текущее число итераций подряд без улучшения целевой функции; Best –
   лучшее решение задачи; cost_best – целевая функция Best.     */
1:  Best = ∅;
2:  cost_best = ∞;
3:  i = 0;
4:  REPEAT
5:      i++;
6:      n_iter = 0;
7:      Построить Curr_i – начальное решение для i-го старта;
8:      REPEAT
9:          Сгенерируем Sol – случайное решение из окрестности
            N_all_better(Curr_i);
10:         n_iter++;
11:         IF (Sol – допустимое решение) THEN
```

```
12:             IF (F(Sol) < F(Curr_i)) THEN
13:                 Curr_i = Sol;
14:                 n_iter = 0;
15:             END IF;
16:         END IF;
17:     UNTIL (n_iter < N_iter_max);
18:     IF (F(Curr_i) < θ × cost_best) THEN // интенсификация поиска
19:         REPEAT
20:             Построить окрестность соседних решений
                N_better(Curr_i);
21:             Найти такое решение z ∈ N_better(Curr_i), что
                F(z) ≤ F(y), для всех y ∈ N_better(Curr_i);
22:             IF (F(z) < F(Curr_i)) THEN
23:                 Curr_i = z;
24:             ELSE
25:                 BREAK;
26:             END IF;
27:         UNTIL (TRUE);
28:     END IF;
29:     IF (F(Curr_i) < cost_best) THEN // обновим лучшее решение
30:         cost_best = F(Curr_i);
31:         Best = Curr_i;
32:     END IF;
33: UNTIL (условие останова не выполнено);
34: RETURN Best;
```

В приложении 2 (рисунок П2.2) приведена блок-схема алгоритма мультистарта для задачи размещения элементов развивающихся информационных систем.

2.7 Решение задачи размещения элементов развивающихся информационных систем при помощи оптимизации подражанием пчелиной колонии

Общие сведения о роевом интеллекте

Методы оптимизации подражанием пчелиной колонии относятся к мультиагентным методам, основанным на моделировании интеллектуального поведения колоний агентов, так называемым методам роевого интеллекта (Swarm Intelligence). В природе подобным интеллектом обладают группы общественных насекомых, например, колонии муравьев, пчел, термитов [8]. Другие названия методов оптимизации подражанием пчелиной колонии: пчелиный алгоритм, метод/алгоритм пчелиного роя, метод/алгоритм пчелиной колонии [53]. В данной диссертации для обозначения данного класса метаэвристик будет использоваться словосочетание "пчелиный алгоритм".

Отметим некоторые особенности роевого интеллекта. Каждый из агентов роя характеризуется стохастическим поведением на основе собственного восприятия окрестности. Агенты руководствуются локальными правилами, не имея никакого представления о глобальной цели. Взаимодействие между подобными самоорганизующимися агентами приводит к появлению коллективного разума под названием роевой интеллект [122]. Особи, входящие в рой, коллективно используют окружающую среду и ресурсы. Ключевой особенностью роевой системы является самоорганизация: низкоуровневые взаимодействия (т.н. микроскопический уровень) приводят к изменениям на глобальном (макроскопическом) уровне [122].

E. Bonabeau [80] под самоорганизацией роя подразумевает следующие 4 особенности:

1) Положительная обратная связь. Например, на основе уровня концентрации феромона, отложенного другими муравьями, муравей выбирает себе путь от

муравейника до источника питания (чем выше уровень феромона на тропе, тем с большей вероятностью муравей выберет эту тропу).

2) Отрицательная обратная связь. Уравновешивает положительную обратную связь и помогает стабилизировать коллективный шаблон поведения. Например, на основе информации от других пчел, пчела может решить, что ее текущий источник нектара значительно хуже других найденных источников, и оставить этот источник [8].

3) Случайность. Флуктуации, такие как случайные блуждания, ошибки, вероятностный поиск очень важны для функционирования роевого интеллекта, т.к. позволяют рою находить новые решения

4) Множественность взаимодействия. Агенты роя используют информацию, поступающую от других агентов, таким образом данные об окружающей среде распространяются по всему рою.

M. Millonas [140], в свою очередь, сформулировал пять принципов, которым должен соответствовать рой, чтобы демонстрировать разумное поведение:

1) Рой должен уметь делать простые пространственные и временные вычисления (принцип близости).

2) Рой должен быть в состоянии реагировать на качественные факторы окружающей среды, такие как качество источников питания или безопасность нахождения (принцип качества).

3) Рой не должен выделять все свои ресурсы по слишком узким каналам, а должен распределять ресурсы между многими узлами (принцип разнообразия отклика).

4) Рой не должен менять режим своего поведения при каждом колебании окружающей среды (принцип стабильности).

5) Рой должен быть в состоянии изменять режим поведения (принцип адаптивности).

Несмотря на то, что особенности самоорганизации, отмеченные E. Bonabeau [80], и принципы, сформулированные M. Millonas [140], четко

прослеживаются в муравьиных колониях и пчелиных роях, решение оптимизационных задач на основе подражания поведению муравьев и пчел – относительная молодая дисциплина. Первые работы, посвященные муравьиному алгоритму, датированы началом 90-х годов Исследования в области пчелиного алгоритма еще моложе: данная концепция получила широкую известность лишь в начале 2000-х.

Биологические основы пчелиного алгоритма

Медоносные пчелы (western honey bees) – это социальные насекомые, которые живут колониями. Существует 3 типа пчел: трутни (drones), рабочие пчелы (workers) и пчелиная матка (queen). Подавляющее большинство разновидностей пчелиного алгоритма основано исключительно на поведении рабочих пчел, т.е. пчел, участвующих в сборе нектара [53].

Источник нектара характеризуется своей полезностью, которая определяется такими факторами, как удалённость от улья, концентрация нектара, удобство его добычи [8].

Рабочие пчелы делятся на 2 типа:

– Занятые фуражиры (employed foragers). Это пчелы, которые добывают нектар из некоторого источника. В теории пчелиных алгоритмов принято, что занятый фуражир в некоторый "привязан" к одному конкретному источнику. Занятый фуражир владеет информацией о полезности своего источника.

– Незанятые фуражиры (unemployed foragers). Эти пчелы которые постоянно ищут новые источники нектара для из дальнейшего использования [123]. Незанятые фуражиры делятся на пчел-разведчиков (они составляют 5-10% от численности всех пчел в улье) и пчел-наблюдателей.

Разведчики (scouts) исследуют окружающую среду в поисках новых источников нектара, а наблюдатели (onlookers) ждут в улье, анализируя информацию о полезности различных источников нектара. По завершении поиска пчела-разведчик возвращается в улей и информирует других членов роя о месте,

количестве и качестве доступных источников питания, которые были ими найдены. Обмен информацией происходит при помощи танца на специально отведенной для этого площадке. Если пчела, наблюдавшая танцы скаутов, решает покинуть улей и собирать нектар, она будет следовать за одним из разведчиков к одному из ранее обнаруженных источников пищи. Такая пчела становится занятым фуражиром. Она занимается сбором нектара, при этом уточняя информацию о количестве нектара в окрестности найденного источника. После сбора фуражир возвращается в улей и оставляет там собранный нектар. Затем он может совершить одно из следующих действий [123]:

– стать незанятым фуражиром, оставив свой текущий источник нектара;

– продолжить добывать нектар из своего источника, не осуществляя вербовку незанятых пчел;

– продолжить добывать нектар из своего источника, выполнив при этом вербовку незанятых пчел.

Описанный процесс продолжается непрерывно, в то время как улей накапливает нектар и исследует новые области с потенциальными источниками питания.

Алгоритм BCO

Алгоритм BCO (Bee Colony Optimization) был предложен авторами D. Teodorović и P. Lućić [135] в начале 2000-х годов и на данный момент является одной из самых популярных концепций роевого интеллекта. Подробное описание алгоритма и его возможных разновидностей приведено в работах [166] и [167]. В данной работе рассматривается модифицированный алгоритм BCO (variant of BCO based on the improvement concept, BCOi) [94]. Особенностью BCOi является то, что в нем рассматривается работа с полными (завершенными) решениями оптимизационной задачи, а не с частичными, как в классическом BCO.

Во всех разновидностях пчелиного алгоритма решение оптимизационной задачи рассматривается как источник нектара. Очевидно, что количество нектара

в источнике пропорционально качеству представляющего его решения. В случае минимизационной задачи полезность источника нектара обратно пропорциональна значению целевой функции решения.

В алгоритме BCOi популяция агентов (искусственных пчел) состоит из B особей, совместно ищущих оптимальное решение. Каждая пчела ответственна за одно решение задачи. Имеется две чередующиеся фазы поиска: прямой проход (forward pass) и обратный проход (backward pass), которые вместе составляют один шаг алгоритма BCO [94,166,167,135].

В процессе прямого прохода каждая пчела исследует пространство поиска. Исследование подразумевает заданное число шагов по улучшению решения. Поиск осуществляется в окрестности текущего решения. В результате исследования получается новое решение. Отметим, что для расширения пространства поиска авторы работ [94,166,167,135] предлагают выбирать не лучшее решение в окрестности, а производить пропорциональную селекцию (т.е. использовать метод рулетки).

После получения новых решений начинается второй этап, обратный проход. В процессе обратного прохода все искусственные пчелы обмениваются информацией о качестве «своих» решений, т.е. о вычисленных значениях целевой функции. Когда обмен информацией полностью завершен, каждая пчела решает (с определенной вероятностью), стоит ли ей отказаться от своего источника нектара. Пчелы, ассоциированные с лучшими решениями, имеют лучшие шансы на то, чтобы сохранить свои текущие решения и «разрекламировать» их среди других пчел. R пчел, которые сохраняют свои решения, в BCOi называются рекрутами. Если пчела отказывается от своего текущего решения, то она должна выбрать одно из решений, предлагаемых другими пчелами. Очевидно, что у лучших решений наибольшая вероятность быть выбранными для дальнейшего исследования.

Две фазы поискового процесса, прямой и обратный проход, чередуются NC раз, т.е. до тех пор, пока каждая пчела не произведет NC модификаций своего

решения. Когда все шаги завершены, определяется лучшее из B имеющихся решений (оно используется для обновления глобального лучшего решения). После этого итерация считается завершенной. Все B решений удаляются, и стартует новая итерация. В качестве критерия останова алгоритма могут выступать, например, максимальное число итераций или суммарное время работы алгоритма.

Вероятность того, что b-я пчела перед началом очередного прямого прохода останется лояльной своему текущему решению (т.е. не откажется от него) выражается формулой:

$$p_b^{u+1} = \exp\left(-\frac{O_{\max} - O_b}{u}\right), \tag{2.8}$$

где O_b – нормализованное значение целевой функции решения, созданного b-й пчелой; O_{max} – максимальное среди всех значений O_b ($b = 1, 2, \ldots, B$); u – порядковый номер прямого прохода в пределах данной итерации ($u = 1, 2, \ldots, NC$).

Нормализованные значения целевой функции рассчитываются следующим образом. Пусть C_b – значение целевой функции для b-й пчелы, C_{max} и C_{min} – соответственно максимальное и минимальное значение целевой функции в пределах улья. Тогда для задачи минимизации [166]:

$$O_b = \frac{C_{\max} - C_b}{C_{\max} - C_{\min}}, \quad b = 1, 2, \ldots, B. \tag{2.9}$$

Для каждой пчелы, которая отказалась от своего решения (их $(B–R)$ штук), мы должны найти рекрута, т.е. выбрать одно из R решений, к которым их пчелы остались лояльны. Вероятность того, что решение данного конкретного рекрута будет выбрано любой из пчел, отказавшихся от своего решения, выражается формулой, соответствующей методу рулетки:

$$p_b = O_b / \sum_{k=1}^{R} O_k, \quad b = 1, 2, \ldots, R. \tag{2.10}$$

В приложении 2 (алгоритм П2.2) приведен псевдокод алгоритма BCOi для задачи минимизации.

Алгоритм ABC

Алгоритм ABC (Artificial Bee Colony) был предложен группой турецких ученых во главе с D. Karaboga [123,124] в середине 2000-х годов и на данный момент является одной из самых популярных концепций роевого интеллекта.

На первом шаге алгоритма ABC случайно генерируется начальная популяция, состоящая из SN решений (источников нектара) [126]. Каждое решение x_i ($i = 1, 2, ..., SN$) представляет собой D-мерный вектор, где D – число оптимизируемых параметров. После инициализации популяция в течение нескольких итераций цикла ($C = 1, 2, ..., MCN$) осуществляет поисковый процесс при помощи занятых фуражиров, пчел-разведчиков и пчел-наблюдателей. Занятый фуражир осуществляет модификацию своего положения (решения) исходя из некой локальной информации (окрестности текущего решения) и проверяет количество нектара (целевую функцию) в новом источнике (новом решении). При условии, что в новом источнике (v_i) нектара больше, чем в предыдущем (x_i), пчела «запоминает» свое новое положение и «забывает» старое. В противном случае, пчела сохраняет в памяти положение предыдущего источника. После того как занятые фуражиры завершают процесс поиска, они обмениваются информацией о концентрации нектара в источниках и их положении с наблюдателями на т.н. области танцев. Пчела-наблюдатель оценивает информацию, полученную от занятых фуражиров, и выбирает источник питания с некоторой вероятностью, пропорциональной количеству нектара в нем. Как и в случае занятого фуражира, наблюдатель производит модификацию положения в своей памяти и проверяет объем нектара источника-кандидата. И опять-таки, при условии, что в новом источнике нектара больше, чем в предыдущем, пчела «запоминает» свое новое положение и «забывает» старое.

Пчела-наблюдатель выбирает i-й источник пищи с некоторой вероятностью p_i, вычисляемой по следующей формуле (формула соответствует методу рулетки):

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}, \tag{2.11}$$

где fit_i – значение целевой функции i-го решения, пропорциональное (для максимизационной задачи) количеству нектара в i-м источнике, а SN – число источников пищи, равное числу занятых фуражиров (BN). Число пчел-наблюдателей также равно BN. Для минимизационной задачи:

$$fit_i = \frac{1}{1 + F_i}. \tag{2.12}$$

Для того чтобы получить положение источника-кандидата в окрестности некоторого текущего источника используется соотношение [125]:

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}), \tag{2.13}$$

где $k \in \{1, 2, ..., SN\}$ и $j \in \{1, 2,...,D\}$ – случайно выбранные индексы. Очевидно, что k не должен быть равен i. φ_{ij} – случайное число в интервале [-1; 1]. Оно отвечает за поиск источников пищи в окрестности x_i.

После того как по формуле (2.13) определены новые источники-кандидаты, происходит выбор между текущим и потенциальным источником на основе жадной эвристики. Т.е. v_i заменяет x_i только если целевая функция v_i лучше.

В алгоритме ABC принято, что если положение пчелы не получается улучшить некоторое заранее определенное число итераций подряд (*limit*), то пчела "отказывается" от своего источника нектара. Его заменяет новый источник, найденный разведчиками.

В приложении 2 (алгоритм П2.3) приведен подробный псевдокод алгоритма ABC для задачи минимизации.

Отличия алгоритмов BCO и ABC

Несмотря на очевидные сходства метаэвристик ABC и BCO, между ними есть ряд существенных различий:

1) Алгоритм ABC предназначен в первую очередь для решения непрерывных задач оптимизации, т.к. формула (2.13) в общем случае применима

только для непрерывных задач. Алгоритм BCO более универсален, т.к. не содержит прямых указаний на то, каким именно образом производить улучшение текущих решений.

2) Пчелы по разным причинам отказываются от своих текущих решений: превышение определенного числа итераций без улучшения решения в ABC и формула (2.8) в BCO.

3) В алгоритме ABC, если пчела отвергла свое решение, ей присваивается новое решение, которое было сгенерировано пчелой-скаутом. В методе BCO в аналогичной ситуации пчеле присваивается решение какой-либо из других пчел.

4) В алгоритме BCO периодически (каждые *NC* шагов) происходит полная «очистка» популяции, т.е. все старые решения удаляются, и генерируются новые *B* решений. В алгоритме ABC одна и та же популяция живет на протяжении всего алгоритма.

5) В алгоритме BCO отсутствует разделение искусственных пчел на различные типы. В алгоритме ABC есть 3 вида пчел: занятые фуражиры, наблюдатели, разведчики.

6) В алгоритме ABC на каждом шаге мы сравниваем лишь одно потенциальное решение v_i с текущим x_i. Если v_i лучше, чем x_i, то v_i становится текущим решением. Т.е. используется эвристика «первое улучшение». В алгоритме BCO просматривается вся окрестность текущего решения.

Можно заметить, что по своей сути пчелиный алгоритм (и ABC, и BCO) очень похож на некий гибридный алгоритм, сочетающий в себе концепции мультистарта и эволюционных алгоритмов.

Пчелиный алгоритм для решения ЗРЭРИС

В данной диссертации предложены два алгоритма оптимального размещения элементов развивающихся информационных систем на основе оптимизации подражанием пчелиной колонии: алгоритм на базе ABC и алгоритм на базе BCO. Отметим, что автору данной диссертации неизвестны работы,

посвященные решению задачи размещения элементов информационных систем при их планировании и оптимизации с использованием алгоритмов ABC и BCO.

В главе 4 при помощи вычислительного эксперимента произведен сравнительный анализ разработанных алгоритмов на базе метаэвристик ABC и BCO.

Алгоритм BCO для решения ЗРЭРИС

В работе [166] рассматривается концепция использования глобальной памяти в алгоритме BCO. Предлагается в конце каждой итерации не удалять все *B* решений, а лучшее из них делать стартовым для одной из пчел на следующей итерации. Такой подход иначе называется концепцией элитаризма. Для удобства обозначим как *use_global* булеву переменную, показывающую, используем ли мы концепцию элитаризма [53].

Отметим еще одну особенность алгоритма BCO. В нем для расширения пространства поиска авторы работы [166] предлагают выбирать не лучшее решение в окрестности, а производить пропорциональную селекцию (использовать метод рулетки). Т.е. вероятность выбора решения пропорциональна его качеству. Однако в работе [78] приводится обоснование возможности использования других методов селекции для пчелиных алгоритмов: турнирной и ранговой селекции, широко применяемых в теории эволюционных алгоритмов. Также в работе [78] приводится описание еще одного метода селекции – дизруптивного. Дизруптивная селекция похожа на пропорциональную, но имеет одно значительное отличие, касающееся формулы, определяющей вероятность выбора особи:

$$p_i = \frac{\left|F_i - F_{avg}\right|}{\sum_{j=1}^{N}\left|F_j - F_{avg}\right|}, \qquad (2.14)$$

где F_{avg} – среднее значение целевой функции для N решений. Дизруптивная селекция направлена на максимизацию вероятности выбора самых лучших и самых худших особей [78]. Особи со «средним» значением целевой функции

имеют не самую высокую вероятность быть выбранными. Считается, что подобная стратегия способствует сохранению разнообразия среди особей роя.

Также метод рулетки используется для определения того, какое решение выберет незанятый фуражир (формула (2.10)). Соответственно, для этой цели также можно использовать любой из 4 методов селекции.

Для поиска новых решений в окрестности некоторого решения *Sol* в алгоритме BCOi будет использоваться окрестность $N_{better}(Sol)$.

В приложении 2 (алгоритм П2.4) представлена общая схема алгоритма BCOi для ЗРЭРИС [53].

Алгоритм ABC для решения ЗРЭРИС

Первоначальная версия алгоритма ABC, описанная в работах [123, 124], подразумевает его использование исключительно для задач непрерывной оптимизации. Однако в ряде новых работ ([163,174] и др.) показано, что ABC также эффективно справляется с решением задач дискретной оптимизации.

В данной диссертационной работе для поиска новых решений в окрестности некоторого решения *Sol* в алгоритме ABC будет использоваться окрестность $N_{better}(Sol)$.

В приложении 2 (алгоритм П2.5) представлена общая схема алгоритма ABC для задачи размещения элементов развивающихся информационных систем [48].

2.8 Решение задачи размещения элементов развивающихся информационных систем при помощи оптимизации подражанием муравьиной колонии

Общие сведения о муравьином алгоритме

Муравьиный алгоритм (алгоритм оптимизации подражанием муравьиной колонии, англ. ant colony optimization, ACO) – один из наиболее эффективных метаэвристических методов, применяющихся для решения NP-трудных задач. ACO относится к классу алгоритмов, основанных на роевом интеллекте. Метод

был предложен M. Dorigo в начале 90-х [101] для решения задачи коммивояжёра (англ. travelling salesman problem, сокращённо TSP). Суть подхода, лежащего в основе ACO, заключается в использовании модели поведения муравьёв, ищущих пути от муравейника к источнику питания. Муравьи используют для передачи информации т.н. стигмергию. Стигмергия — это разнесенный во времени тип взаимодействия, когда один субъект взаимодействия изменяет некоторую часть окружающей среды, а остальные используют информацию об ее состоянии позже, когда находятся в ее окрестности [71]. Две основные характеристики, отличающие стигмергию от других способов коммуникации, заключаются в следующем [97]:

– стигмергия представляет собой косвенную, несимволическую форму общения за счёт изменения насекомыми окружающей среды;

– информация, передаваемая посредством стигмергии, является локальной, т.е. она доступна только тем насекомым, которые находятся в месте, где другое насекомое оставило некую стигмергическую субстанцию, или в его непосредственной окрестности.

Биологически стигмергия реализуется через феромон – специальный секрет, который муравей откладывает на тропе в процессе своего перемещения. Другие муравьи воспринимают присутствие феромонов и имеют тенденцию следовать по пути, где концентрация феромона выше [98]. Благодаря этому механизму, муравьи в состоянии находить источники еды удивительно эффективным способом. Феромон с течением времени постепенно испаряется, что позволяет муравьям адаптировать свое поведение под изменения внешней среды. Распределение феромона по пространству передвижения муравьев является своего рода динамически изменяемой глобальной памятью муравейника [71].

В методе ACO имитируется деятельность колонии искусственных муравьев. Муравьи взаимодействуют между собой для решения поставленной оптимизационной задачи посредством обмена информацией через феромон. Для решения задачи при помощи муравьиного алгоритма ее удобно представить в

виде модели на графе. Например, в задаче коммивояжёра города представляют собой вершины графа, а рёбра *(i, j)* между вершинами *i* и *j* — пути сообщения между этими городами. С каждым ребром связан его феромонный уровень τ_{ij} и эвристическая функция η_{ij} (выражает желание посетить город *j* из города *i*), обратно пропорциональная длине пути между городами *i* и *j*.

Муравьиный алгоритм строит решение из комбинации уникальных компонент, выбираемых из конечного набора. Целью является поиск оптимальной комбинации компонент. Выбор компоненты решения (в случае TSP – ребра графа c_{ij}) из множества допустимых компонент осуществляется вероятностно на каждом шаге построения решения, исходя из значений феромонного уровня и эвристической функции ребер графа.

Одна итерация алгоритма подразумевает построение каждым из *A* муравьев колонии одного допустимого решения задачи. Феромонный уровень компонент периодически обновляется согласно следующему правилу: феромон ребра получает приращение, пропорциональное качеству (в случае минимизационной задачи качество обратно пропорционально значению целевой функции) решения, в которое это ребро входит.

Существует множество разновидностей ACO. В данной работе для решения ЗРЭРИС будут использованы три наиболее известных подхода: Ant System (AS) [99,101], MAX–MIN Ant System (MMAS) [162] и Ant Colony System (ACS) [100].

Муравьиный алгоритм для решения ЗРЭРИС

Введем несколько обозначений для краткости. В дальнейшем парой *(s, t)* применительно к решению нашей задачи будем называть элемент *t*-го типа, установленный на *s*-м месте-кандидате. Фраза "пара *(s, t)* используется в решении" означает, что $y_{st} = 1$. Триплетом *(i, s, t)* применительно к решению нашей задачи будем называть подключение *i*-го клиента к элементу *t*-го типа, установленному на *s*-м месте-кандидате. Фраза "триплет *(i, s, t)* используется в решении" означает, что $y_{st} = 1$ и $x_{is} = 1$.

В данной диссертационной работе предлагается алгоритм ACO, отличительной особенностью которого является использование двух колоний (а не одной, как в классическом алгоритме) искусственных муравьев.

Впервые концепция использования нескольких колоний муравьев (MACO, Multiple Ant Colony Optimization) была предложена в работе [109]. Концепция MACO используется, как правило, для задач мнокритериальной оптимизации, когда каждая из колоний муравьев отвечает за одну из целевых функций [4,109]. Однако в работе [168] предлагается подход, согласно которому MACO можно использовать и для задач однокритериальной оптимизации.

Работы [18,28] посвящены решению задач размещения с использованием одной колонии муравьев. Отметим работы, посвященные решению задач размещения с использованием нескольких колоний муравьев. Работа [49] посвящена решению задачи SSCFLP при помощи мультиколониального алгоритма ACO. В статье [155] рассматривается частный случай SSCFLP – задача размещения базовых станций при планировании беспроводных сетей передачи данных. Работа [54] посвящена ЗРЭРИС как частному случаю проектирования распределенной информационной системы.

Возможность использования двух колоний муравьев для решения ЗРЭРИС обусловлена следующими соображениями. Как уже было сказано ранее, для возможности применения ACO необходимо обеспечить представление задачи в виде графа. На рисунке 2.1 приведено схематичное представление готового решения ЗРЭРИС (для частного случая ИС – беспроводной сети передачи данных). Кружками обозначены места-кандидаты. Красные кружки – это места-кандидаты, на которые был установлен элемент ИС. Зеленые квадратики – это клиенты. Соединительная линия обозначает то факт, что клиент подключен к элементу. Числа внутри кружков обозначают номер места-кандидата.

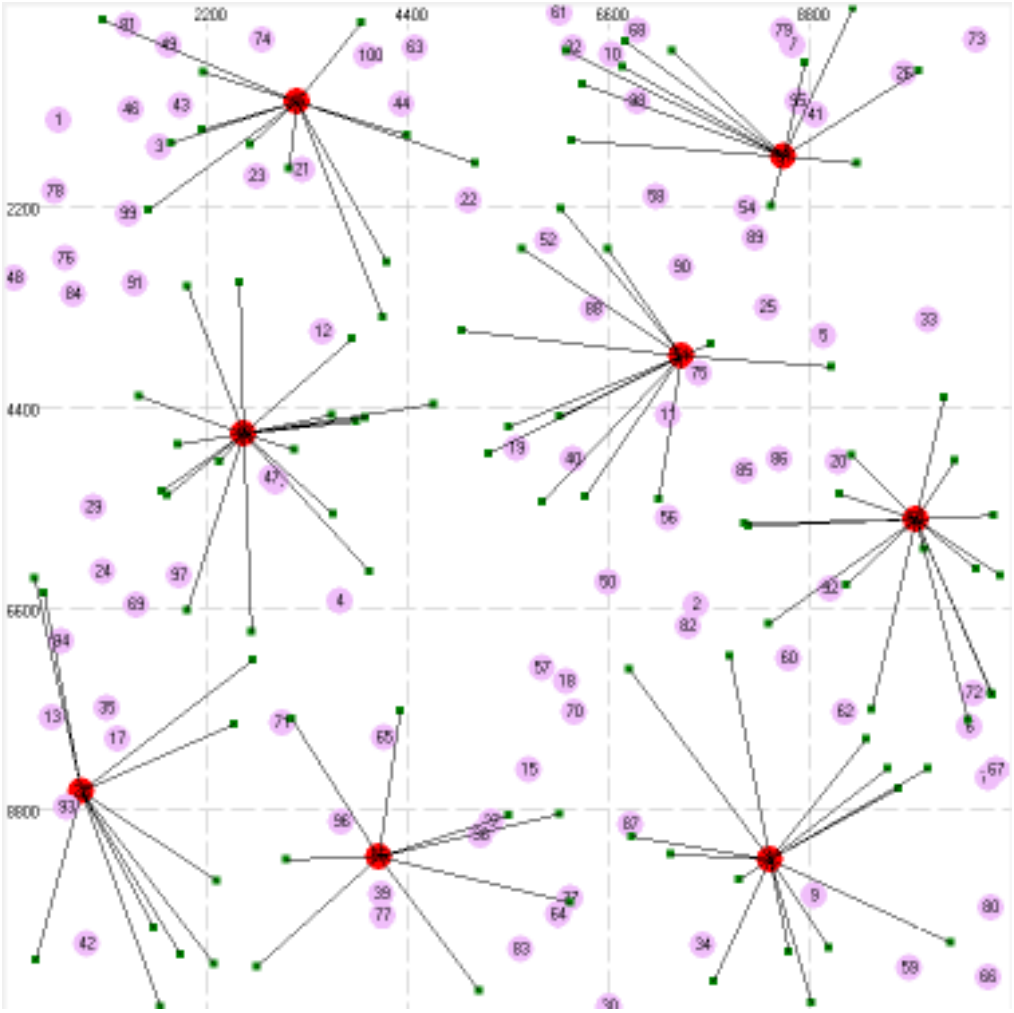

Рисунок 2.1 – Схематичное представление готового решения ЗРЭРИС

В ЗРЭРИС множество вершин V_1 соответствует местам-кандидатам на размещение, множество вершин V_2 представляет клиентов, а множество ребер E ассоциировано с фактом подключения клиентов к элементам ИС. Таким образом, граф, на котором конструируется решение задачи, состоит из вершин двух типов и множества ребер, соединяющих вершины разных типов между собой, т.е. является двудольным графом [54]. При этом места-кандидаты между собой не соединены ребрами, точно так же, как и между клиентами отсутствуют ребра, ведь эти связи не имеют смысла (в контексте нашей задачи) [155].

Отметим, что концепции, используемые при решении задачи коммивояжера, не применимы для нашей задачи. Ведь в задаче коммивояжера решение строится из компонент всего одного типа (ребер графа). В нашей же задаче решение строится из двух типов компонент. Это и обуславливает

целесообразность использования двух муравьиных колоний (размер каждой колонии составляет *A* муравьев) [47,54].

Нахождение решения задачи будет осуществляться в два этапа. Первая колония отвечает за выбор элементов. Пусть τ_1 – двумерный массив, каждый его элемент $\tau_1[s][t]$ хранит уровень феромона для пары (s, t). Определим эвристическую функцию для первой колонии. Значение эвристической функции некоторой компоненты должно быть пропорционально «желанию» выбрать эту компоненту исходя из её свойств. Для каждой компоненты решения мы можем посчитать $\eta_1[s][t]$ заранее, т.к. значение эвристической функции (в отличие от феромона) не меняется в процессе работы алгоритма. Эвристическую функцию места-кандидата с установленным элементом определим как отношение производительности элемента к сумме его цены и стоимости его установки на место:

$$\eta_1[s][t] = prf_t / (ElCost_t + PsCost_s). \quad (2.15)$$

Вторая колония отвечает за подключение клиентов к выбранным ранее элементам. Пусть τ_2 – трехмерный массив, каждый его элемент $\tau_2[i][s][t]$ хранит уровень феромона ребра, соединяющего *i*-го клиента с *s*-м местом, если на нем установлен элемент *t*-го типа. Определим эвристическую функцию для второй колонии:

$$\eta_2[i][s][t] = 1/Q(i,s,t), \quad (2.16)$$

где *Q(i, s, t)* – функция, для каждого триплета *(i, s, t)*, обратно пропорциональная уровню межмодульных помех при подключении *i*-го клиента к элементу *t*-го типа, установленному на *s*-м месте.

Таким образом, две колонии строят решение из разных типов компонентов. Для 1-й колонии компонентами решения являются пары (s, t) ($\forall\ s \in \{1, 2, \ldots, N_{ps}\}$, $\forall\ t \in \{1, 2, \ldots, N_{types}\}$). Для 2-й колонии компонентами решения являются триплеты (i, s, t) ($\forall\ i \in \{1, 2, \ldots, N_{cl}\}$, $\forall\ s \in \{1, 2, \ldots, N_{ps}\}$, $\forall\ t \in \{1, 2, \ldots, N_{types}\}$).

Выбор мест-кандидатов. Отметим, что перед тем как муравей 1-й колонии начинает свою работу, происходит случайная генерация типа элемента для

каждого места. Подобное действие предназначено для расширения пространства поиска.

В действительности, первая колония муравьев не формирует множество используемых мест, а лишь ранжирует места с установленными на них элементами в порядке убывания «желанности». Муравей из второй колонии будет поступать с проранжированными элементами следующим образом: сначала клиенты будут подключаться к первому элементу, пока не перестанет выполняться ограничение (1.14), потом по той же схеме ко второму и т.д.; работа муравья из второй колонии закончится тогда, когда все клиенты будут подключены к своим элементам. Соответственно, в результирующее множество выбранных мест войдут те места, к которым подключен хотя бы один клиент.

Ранжирование мест происходит следующим образом. Муравей h из 1-й колонии последовательно должен N_{ps} раз выполнить следующие шаги. Из множества O^h еще непроранжированных мест-кандидатов с установленными на них элементами выбрать самое «желанное», пусть его номер s. Выбор места s для метода ACS происходит согласно формулам (2.17) и (2.18), для AS и MMAS только согласно формуле (2.18). P_s^h – вероятность того, что h-й муравей выберет s-е место из множества O^h еще непроранжированных этим муравьем мест. α_1 – параметр, определяющий относительную значимость феромонного уровня места с установленным элементом определенного типа; β_1 – параметр, определяющий относительную значимость эвристической функции η_1. q^1 – случайное число, равномерно распределенное в интервале [0; 1], q_0^1 – коэффициент исследования для 1-й колонии.

$$s = \begin{cases} \arg\max\limits_{s \in O^h} \left((\tau_1[s][t])^{\alpha_1} (\eta_1[s][t])^{\beta_1} \right), & \text{если } q^1 \le q_0^1. \\ S, & \text{иначе.} \end{cases} \tag{2.17}$$

$$S: \quad P_s^h = \frac{(\tau_1[s][t])^{\alpha_1} (\eta_1[s][t])^{\beta_1}}{\sum\limits_{j \in O^h} (\tau_1[j][t])^{\alpha_1} (\eta_1[j][t])^{\beta_1}}. \tag{2.18}$$

Подключение клиентов. Выбор подключаемого клиента для метода ACS происходит согласно формулам (2.19) и (2.20), для AS и MMAS только согласно формуле (2.20). P_{si}^h – вероятность того, что h-й муравей выберет i–й клиент из множества W^h клиентов, которые еще не подключены, для подключения к s-му элементу. Отметим, что i-й клиент мы можем подключить к s-му месту с t-м типом элемента только, если для подобного подключения выполняются ограничения (1.15).

α_2 – параметр, определяющий относительную значимость феромонного уровня ребер, соединяющих элемент с клиентом; β_2 – параметр, определяющий относительную значимость эвристической функции η_2. q^2 – случайное число, равномерно распределенное в интервале [0; 1], q_0^2 – коэффициент исследования для 2-й колонии.

Элементы ИС перебираются в том порядке, в котором их выбрал муравей из 1-й колонии: клиенты подключаются к первому элементу, пока не перестанет выполняться ограничение на производительность оборудования (1.14), затем к следующему и т.д., до тех пор пока множество W^h не станет пустым.

$$i = \begin{cases} \arg\max\limits_{i \in W^h} \left((\tau_2[s][t])^{\alpha_2} (\eta_2[s][t])^{\beta_2} \right), & \text{если } q^2 \le q_0^2. \\ I, & \text{иначе.} \end{cases} \tag{2.19}$$

$$I: \quad P_{si}^h = \frac{(\tau_2[i][s][t])^{\alpha_2} (\eta_2[i][s][t])^{\beta_2}}{\sum\limits_{z \in W^h} (\tau_2[z][s][t])^{\alpha_2} (\eta_2[z][s][t])^{\beta_2}}. \tag{2.20}$$

Таким образом, одно решение получается в результате последовательной работы сначала h-го муравья 1-й колонии, а затем h-го муравья второй колонии.

Обновление феромона. Феромонные уровни муравьев 1-й колонии и муравьев 2-й колонии будут обновляться независимо.

1) Ant System method. В AS предусмотрен один тип обновления феромонного уровня компонент решения – глобальное обновление. Оно происходит в конце каждой итерации алгоритма, когда все A муравьев обеих колоний закончили свою работу, и мы имеем A готовых решений. Особенностью

обновления феромона в алгоритме AS является то, что феромонный уровень компонент обновляют все *A* муравьев, построивших решения.

$$\tau_1[s][t] = (1-\rho_1)\cdot\tau_1[s][t] + \sum_{h=1}^{A}\Delta\tau_{st}^{h}. \tag{2.21}$$

$$\Delta\tau_{st}^{h} = \begin{cases} 1/F_h, & \text{если } (s,t)\in Sol_h. \\ 0, & \text{иначе,} \end{cases} \tag{2.22}$$

где $\rho_1 \in (0; 1]$ – скорость испарения феромонного следа муравьев 1-й колонии; F_h – значение целевой функции решения Sol_h, найденного h-ми муравьями 1-й и 2-й колонии.

$$\tau_2[i][s][t] = (1-\rho_2)\cdot\tau_2[i][s][t] + \sum_{h=1}^{A}\Delta\tau_{ist}^{h}. \tag{2.23}$$

$$\Delta\tau_{st}^{h} = \begin{cases} 1/F_h, & (i,s,t)\in Sol_h. \\ 0, & \text{иначе,} \end{cases} \tag{2.24}$$

где $\rho_2 \in (0; 1]$ – скорость испарения феромонного следа муравьев 2-й колонии.

2) MAX–MIN Ant System. Данный алгоритм имеет два отличия по сравнению с AS. Во-первых, значение феромонного уровня ограничено сверху и снизу. Во-вторых, только лучший муравей (т.е. муравей, построивший лучшее решение) обновляет феромон компонент, входящих в его решение.

$$\tau_1[s][t] = \left[(1-\rho_1)\cdot\tau_1[s][t] + \Delta\tau_{st}^{best}\right]_{\tau_{1min}}^{\tau_{1max}}. \tag{2.25}$$

$$\Delta\tau_{st}^{best} = \begin{cases} 1/F_{best}, & \text{если } (s,t)\in Sol_{best}. \\ 0, & \text{иначе,} \end{cases} \tag{2.26}$$

где F_{best} – значение целевой функции лучшего решения на данной итерации Sol_{best}; τ_{1max} и τ_{1min} – соответственно максимальное и минимальное значение феромонного уровня для 1-й колонии. Оператор $[x]_b^a$ может быть описан так:

$$[x]_b^a = \begin{cases} a, & \text{если } x > a. \\ b, & \text{если } x < b. \\ x, & \text{иначе.} \end{cases} \tag{2.27}$$

Для 2-й колонии:

$$\tau_2[i][s][t] = \left[(1-\rho_2)\cdot\tau_2[i][s][t] + \Delta\tau_{ist}^{best}\right]_{\tau_{2min}}^{\tau_{2max}}. \quad (2.28)$$

$$\Delta\tau_{ist}^{best} = \begin{cases} 1/F_{best}, & \text{если } (i,s,t) \in Sol_{best}. \\ 0, & \text{иначе,} \end{cases} \quad (2.29)$$

где τ_{2max} и τ_{2min} – соответственно максимальное и минимальное значение феромонного уровня для 2-й колонии.

Как правило, величины τ_{1max}, τ_{1min}, τ_{2max}, τ_{2min} жестко задаются перед началом работы алгоритма, и их значения подбираются с учетом специфики задачи. Однако авторы алгоритма MMAS советуют их пересчитывать после каждой итерации алгоритма по следующим формулам [162]:

$$\tau_{1max} = \frac{1}{1-\rho_1}\cdot\frac{1}{F_{best}}. \quad (2.30)$$

$$\tau_{1min} = \frac{\tau_{1max}(1-\sqrt[n]{p_{best}})}{\sqrt[n]{p_{best}}(avg-1)}, \quad (2.31)$$

где $avg = n/2$ (n – размерность задачи, в случае ЗРЭРИС равная $N_{cl}\times N_{ps}\times N_{types}$); p_{best} – вероятность того, что лучшее решение задачи уже найдено. Рекомендуется выбирать p_{best} в диапазоне [0.01; 0.05]. Формулы для нахождения τ_{2max}, τ_{2min} аналогичны формулам (2.30) и (2.31).

3) Ant Colony System. В данном алгоритме в отличие от AS и MMAS помимо глобального обновления феромона происходит т.н. локальное обновление. Оно происходит не в конце каждой итерации: как только h-й муравей построил решение, он обновляет локальный уровень феромона компонент, использованных им в решении. Для 1-й колонии:

$$\tau_1[s][t] = (1-\varphi_1)\cdot\tau_1[s][t] + \varphi_1\tau_{10}. \quad (2.32)$$

Локальное обновление для компонент муравья 2-й колонии:

$$\tau_2[i][s][t] = (1-\varphi_2)\cdot\tau_2[i][s][t] + \varphi_2\tau_{20}, \quad (2.33)$$

где $\varphi_1 \in (0; 1]$ и $\varphi_2 \in (0; 1]$ – коэффициент затухания феромона соответственно для 1-й и 2-й колонии; τ_{10} и τ_{20} – начальные значения феромонного уровня соответственно для 1-й и 2-й колонии.

Глобальное обновление феромона:

$$\tau_1[s][t] = \begin{cases} (1-\rho_1)\tau_1[s][t] + \rho_1 \cdot \Delta\tau_{st}^{best}, & \text{если } (s,t) \in Sol_{best}. \\ \tau_1[s][t], & \text{иначе.} \end{cases} \quad (2.34)$$

$$\tau_2[i][s][t] = \begin{cases} (1-\rho_2)\tau_2[i][s][t] + \rho_2 \cdot \Delta\tau_{ist}^{best}, & \text{если } (i,s,t) \in Sol_{best}. \\ \tau_2[i][s][t], & \text{иначе.} \end{cases} \quad (2.35)$$

Ниже представлена общая схема муравьиного алгоритма для задачи размещения элементов развивающихся информационных систем [49,47,155,54].

Алгоритм 2.11 Псевдокод муравьиного алгоритма для ЗРЭРИС

```
/* Введем обозначения: Sol_glob – лучшее известное решение задачи, C_glob –
   стоимость Sol_glob, Sol_iter – лучшее решение задачи на данной итерации
   алгоритма, C_iter – стоимость Sol_iter. */
Инициализировать параметры α1, α2, β1, β2, ρ1, ρ2, φ1, φ2, A;
Расчет массивов η1 и η2 согласно формулам (2.15) и (2.16);
Инициализация массивов τ1 и τ2;
C_glob = ∞;
C_iter = ∞;
FOR h = 1 TO A DO
    Для каждого места-кандидата случайно генерируем тип
    элемента в диапазоне [1; N_types];
    Выбор мест-кандидатов h-м муравьем 1-й колонии согласно
    формулам (2.17) и (2.18);
    Подключение клиентов h-м муравьем 1-й колонии согласно
    формулам (2.19) и (2.20);
    // На данный момент у нас сформировано решение Sol_h;
    IF (используется алгоритм ACS) THEN
        Локальное обновление феромонного уровня согласно
        формулам (2.32) и (2.33);
    END IF;
```

```
13:     IF (C_iter > F(Sol_h)) THEN
14:         Sol_iter = Sol_h;
15:         C_iter = F(Sol_h);
16:     END IF;
17: END FOR;
18: IF (C_glob > F(Sol_iter)) THEN
19:     Sol_glob = Sol_iter;
20:     C_glob = F(Sol_iter);
21: END IF;
22: Глобальное обновление феромонного уровня;
23: IF (условие останова не выполнено) THEN
24:     переход к команде 6;
25: END IF;
26: RETURN Sol_glob;
```

В приложении 2 (рисунок П2.3) приведена блок-схема муравьиного алгоритма для задачи размещения элементов развивающихся информационных систем.

2.9 Выводы

1. Разработан эволюционный алгоритм для интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем. Алгоритм сформулирован так, что в зависимости от заданных параметров может представлять собой различные разновидности ЭА: генетический алгоритм и эволюционные стратегии.

2. Сформулированы 6 операций, которые можно применить к текущему решению ЗРЭРИС, чтобы получить новое решение из окрестности: «Смена типа одного элемента на более дешевый», «Смена типа одного элемента на более

дорогой», «Переподключение одного клиента», «Удаление одного элемента», «Добавление одного элемента», Перемещение одного элемента».

3. Разработан алгоритм локального спуска для интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем.

4. Разработан алгоритм имитации отжига для интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем.

5. Разработан алгоритм вероятностного поиска с запретами для интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем. Используется следующий подход: в список запретов добавляются не конкретные прошлые решения, а операции по изменению конфигурации информационной системы, которые могут вернуть нас в старые локальные оптимумы.

6. Разработан алгоритм мультистарта для интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем. В предлагаемом алгоритме используется жадная эвристика «первое улучшение».

7. Разработан алгоритм интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем на основе оптимизации подражанием муравьиной колонии. Отличительной особенностью предлагаемого алгоритма является использование двух колоний искусственных муравьев.

8. Разработаны алгоритмы интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем на основе оптимизации подражанием пчелиной колонии (алгоритмы ABC и BCO).

3 МЕТОД НАСТРОЙКИ УПРАВЛЯЮЩИХ ПАРАМЕТРОВ МЕТАЭВРИСТИЧЕСКИХ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ РАЗМЕЩЕНИЯ ЭЛЕМЕНТОВ РАЗВИВАЮЩИХСЯ ИНФОРМАЦИОННЫХ СИСТЕМ

3.1 Системный анализ процесса настройки управляющих параметров для метаэвристических алгоритмов оптимизации

В главе 2 были описаны метаэвристические алгоритмы интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем. Встает вопрос оценки качества функционирования предложенных алгоритмов. Всякое решение, найденное некоторым оптимизационным алгоритмом, может быть оценено с точки зрения двух характеристик:

– время нахождения решения;

– значение целевой функции найденного решения.

При этом нельзя точно сказать, какое именно сочетание этих характеристик является желаемым. Очевидно, что идеальным случаем является нахождение точного оптимального решения за крайне малый промежуток времени. Однако для NP-трудных задач это является невозможным. Сама суть NP-трудности подразумевает экспоненциальное время нахождения точного решения методом полного перебора. Поэтому нахождение решения при помощи метаэвристик подразумевает компромисс между скоростью нахождения и точностью полученного решения.

Параметры метаэвристических алгоритмов, которые может задавать лицо, принимающее решение, называются управляющими параметрами [141] или свободными параметрами алгоритма. В работе [22] вектор (набор) значений свободных параметров алгоритма называется стратегией алгоритма.

Для каждой метаэвристики верно высказывание: разные наборы управляющих параметров алгоритма обеспечивают получение решений разного качества.

Мы можем сравнивать между собой качество решений, получаемых при помощи разных метаэвристических алгоритмов. Также мы можем сравнивать решения, получаемые одним и тем же алгоритмом, но с разным набором управляющих параметров.

Чтобы сравнение было корректным, мы должны пойти одним из двух путей:

– Зафиксировать время, отводимое на решение. Тогда сравнивать мы будем значения целевых функций найденных решений.

– Зафиксировать значения целевых функций получаемых решений. Тогда сравнивать необходимо время, затраченное алгоритмом на получение подобного решения.

Первый способ, во-первых, легче в реализации. А во-вторых, фиксация желаемого значения целевой функции может привести к ситуации, когда алгоритм будет работать крайне долго, что нежелательно, т.к. для корректной настройки необходим многократный запуск алгоритма. Обобщая вышесказанное, настраивать значения параметров алгоритма, сравнивая решения, получаемые одним и тем же алгоритмом при разных наборах свободных параметров, желательно следующим образом:

– Необходимо сформулировать тестовую задачу (или несколько задач), для решения которой мы планируем подбирать параметры. Например, параметры алгоритма, подобранные для задач малой и средней размерности (5×5×2, 7×7×3, 10×10×3 и т.д.) могут давать не самый лучший результат для задач средней и большой размерности (50×50×3, 100×100×2, 500×500×3 и т.д.).

– Необходимо зафиксировать время, которое будет отводиться на решение задачи при каждом запуске алгоритма.

– Для каждого набора параметров произвести несколько запусков исследуемого алгоритма.

– Оценить полученные результаты с целью определения лучшего набора параметров.

Существует два возможных подхода к решению задачи оптимизации параметров метаэвристических алгоритмов [165,22]:

– офлайн-оптимизация;

– онлайн-оптимизация.

При онлайн-оптимизации значения управляющих параметров изменяются непосредственно в процессе решения задачи. Большинство алгоритмов, реализующих онлайн-оптимизацию используют либо динамический, либо самоадаптивный подход [165, 84]. Очевидным недостатком оптимизации непосредственно в процессе поиска является неуниверсальность подобного подхода. Например, в нашем случае, требовалось бы придумать модификации для каждого из предложенных в главе 2 алгоритмов. Стратегии самоадаптации параметров, применимые, например, для эволюционных алгоритмов, не подходят для класса алгоритмов локального поиска и т.д.[165]. Классическим примером является динамическая метаоптимизация значения вероятности мутации для эволюционного алгоритма. В ее основе лежит представление о том, что на ранних шагах алгоритма вероятность должна быть достаточно большой (для широты пространства поиска), постепенно снижаясь в процессе функционирования алгоритма (для получения достаточно точного решения). Таким образом, для корректного проведения динамической адаптации необходимо иметь, по крайней мере, приблизительное представление о том, как каждый параметр должен изменяться в процессе функционирования алгоритма.

При офлайн-оптимизации управляющих параметров значения различных параметров фиксируются до начала выполнения метаэвристики. В результате ряда экспериментов лучший из наборов параметров выбирается в качестве результата настройки. Главным недостатком офлайн-оптимизации являются высокие требования ко времени, требующемуся на процедуру настройки параметров.

Плюсом же является универсальность: грамотный метод офлайн-оптимизации подразумевает свое применение для нескольких метаэвристических алгоритмов.

Целью главы 3 является создание универсального метода офлайн-оптимизации значений параметров метаэвристик, предложенных в главе 2, основывающегося на эволюционном подходе.

Теоретически возможно использовать для решения задачи настройки параметров полный перебор. Однако если у нас есть n параметров, каждый из которых может принимать k значений, нам необходимо будет выполнить n^k экспериментов. Т.е. задача настройки управляющих параметров является экспоненциально сложной.

Проблема нахождения параметров некоторого метаэвристического алгоритма может быть сформулирована как оптимизационная задача. А так как сами метаэвристические алгоритмы являются алгоритмами оптимизации, то офлайн-оптимизация параметров метаэвристик может быть названа «метаоптимизацией» [165]. Более того, так как метаоптимизация может быть выполнена при помощи метаэвристик, подход, применяемый в данной работе, можно назвать «метаметаэвристическим».

Схема метаоптимизации приведена на рисунке 3.1. Метаоптимизация подразумевает два уровня: метауровень и базовый уровень. На метауровне метаэвристика работает с решениями, представляющими собой параметры метаэвристик базового уровня, которые необходимо оптимизировать. Решение x_i на метауровне представляет собой (кодирует) конкретные значения параметров, оптимизируемых при помощи метаэвристик базового уровня (например, длина списка запретов, параметр рандомизации окрестности и др. в поиске с запретами). На метауровне целевая функция $F\_meta(X)$ ассоциирована с лучшим из решений, найденных на базовом уровне при помощи метаэвристик с параметрами, кодируемыми решениями x_i ($i = 1, 2, \ldots, n\_set$). X – множество всех возможных наборов параметров; n_set – общее число различных наборов параметров, мощность множества X. Метаэвристики на базовом уровне работают с

решениями, которые являются непосредственными решениями исходной оптимизационной проблемы (в нашем случае – ЗРЭРИС). Целевая функция *F_base* используется метаэвристиками базового уровня для обозначения целевой функции конкретного решения при конкретном наборе параметров x_i ($i = 1, 2, \ldots, n\_set$). Для минимизационной задачи справедлива формула

$$F\_meta(X) = \min_{i=1,\ldots,n\_set} F\_base(x_i). \quad (3.1)$$

Окончательным решением метаоптимизационной задачи будет набор параметров x_{best_set}:

$$best\_set = \arg\min_{i=1,\ldots,n\_set} F\_base(x_i). \quad (3.2)$$

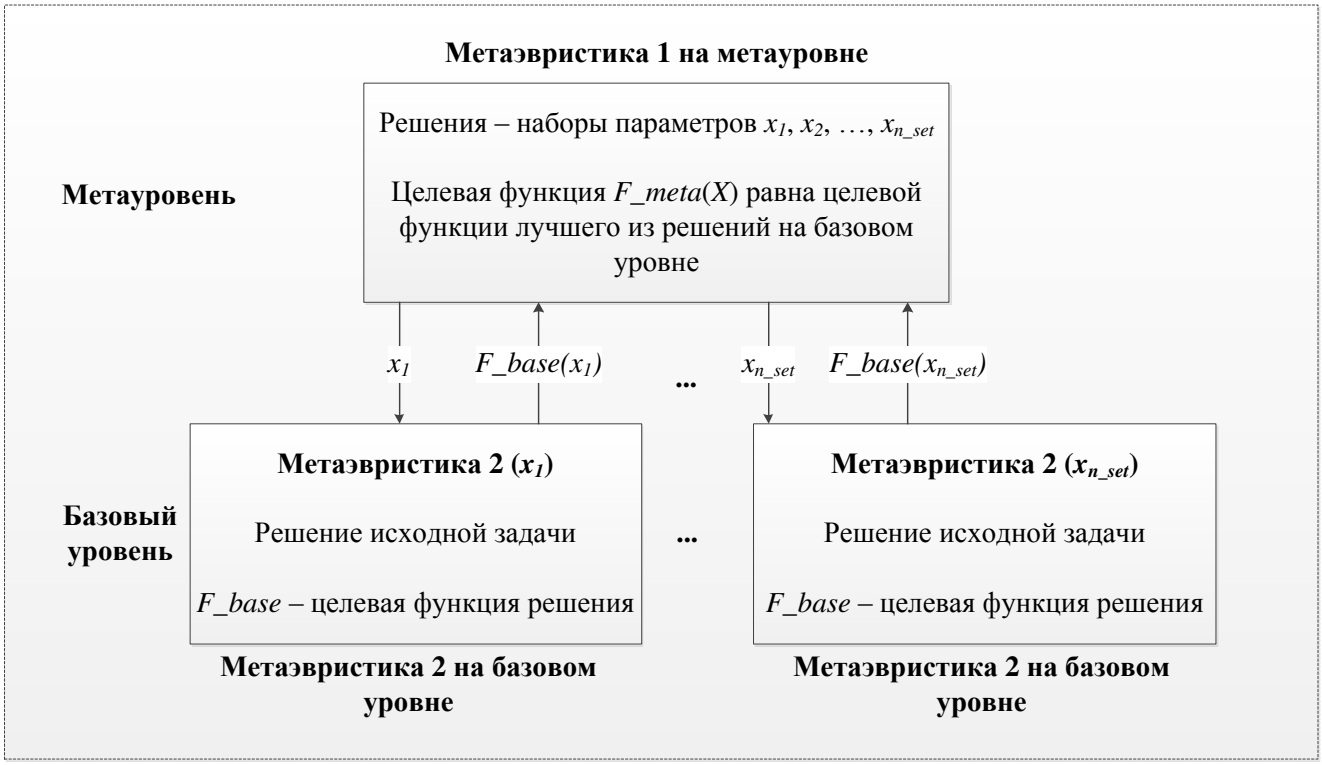

Рисунок 3.1 – Метаоптимизация с использованием метаметаэвристического подхода

Далее в данной работе будет описан эволюционный метод метаоптимизации значений параметров некоторого алгоритма решения ЗРЭРИС. Если метод метаоптимизации рассматривать как систему, то с точки зрения классификации [63,69,3] ее можно назвать:

– абстрактной;

– математической;

– многоэлементной;
– гетерогенной;
– динамической.

На рисунке 3.2 схематично изображен процесс метаоптимизации некоторого алгоритма. Лицо, принимающее решения, задает набор (множества) возможных значений свободных параметров исходного базового алгоритма. Метод метаоптимизации принимает этот набор в качестве входа, а по окончании вычислений выдает вектор значений, с использованием которых базовый алгоритм показывает наилучший результат, лицу, принимающему решения. ЛПР на основе анализа результата метода метаоптимизации имеет возможность запустить вычисления еще раз, передав на вход уже другой набор возможных значений управляющих параметров.

Сам метод метаоптимизации представляет собой двухкомпонентную систему (см. рисунок 3.3). В него входят базовый оптимизируемый алгоритм решения ЗРЭРИС и блок эволюционных вычислений, собственно и производящий метаоптимизацию. Блок эволюционных вычислений передает на вход базовому алгоритму вектор значений параметров. В качестве выхода алгоритм решения ЗРЭРИС выдает решение ЗРЭРИС (его целевую функцию), полученное с использованием входного вектора параметров. Блок эволюционных вычислений ответственен за оценку решений, выданных алгоритмом решения ЗРЭРИС, формирование популяции в каждом поколении алгоритма и за последовательную передачу на вход базовому алгоритму всех особей популяции.

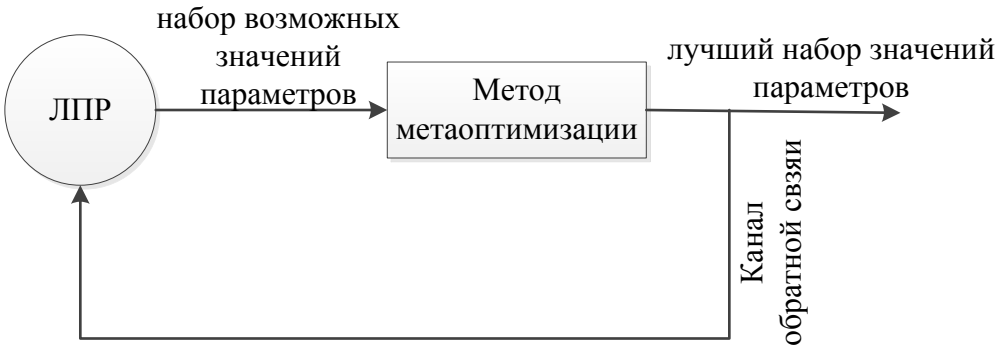

Рисунок 3.2 – Процесс принятия решений при метаоптимизации

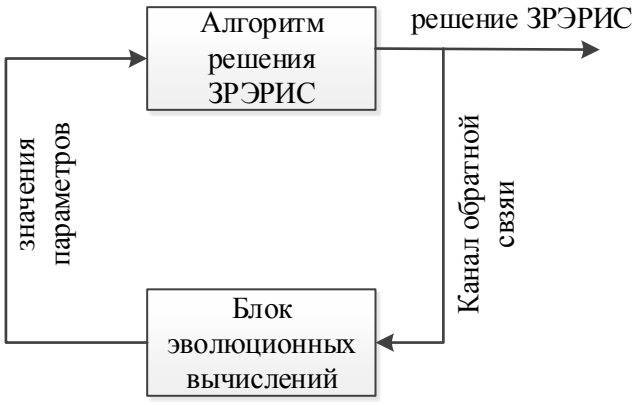

Рисунок 3.3 – Метод метаоптимизации

3.2 Применение эволюционного подхода для оптимизации параметров разработанных метаэвристик

Поставим проблему следующим образом. У нас есть ЗРЭРИС размерности $N_{cl} \times N_{ps} \times N_{types}$. Мы пытаемся решить ее при помощи некоторого алгоритма.

Функционирование алгоритма определяется набором из *npar* переменных управляющих параметров: $\{\chi_1, \chi_2, \ldots, \chi_{npar}\}$. Каждый из параметров является элементом некоторого конечного множества: $\chi_1 \in A_1$, $\chi_2 \in A_2$, …, $\chi_{npar} \in A_{npar}$. A_1, A_2, …, A_{npar} – множества возможных значений параметров, из которых будет происходить выбор, известны заранее [29].

Каждый набор параметров конкретного алгоритма может быть представлен в виде хромосомы (рисунок 3.4).

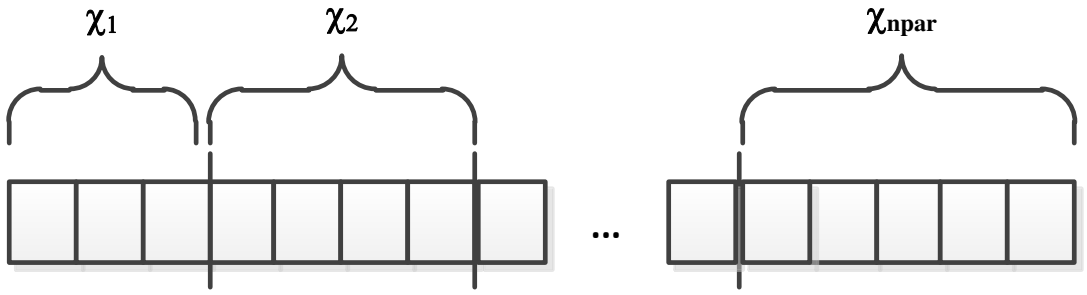

Рисунок 3.4 – Представление параметров алгоритма в виде хромосомы

Каждый «квадратик» (бит) на рисунке 3.4 может принимать значение 0 или 1. Хромосома состоит из генов. В качестве генов в данном случае выступают не

отдельные биты, а целые участки хромосом (на рисунке 3.4 они разделены вертикальной чертой). Каждый ген кодирует один из параметров алгоритма. Ген, представляя собой последовательность битов, является двоичным числом. Обозначим эти числа следующим образом: z_1, z_2, …, z_{npar}. Так как каждое из множеств A_1, A_2, …, A_{npar} является конечным, мы можем рассматривать любое из этих множеств как одномерный массив (вектор) [52]. Соответственно гены z_1, z_2, …, z_{npar} кодируют позицию параметров χ_1, χ_2, …, χ_{npar} в массивах A_1, A_2, …, A_{npar}.

Приведем пример. Пусть хромосома имеет вид (рисунок 3.5).

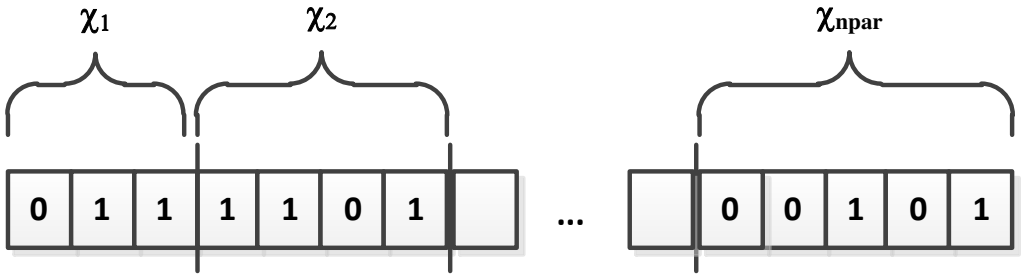

Рисунок 3.5 – Пример хромосомы

Тогда $z_1 = 011_2$, $z_2 = 1101_2$, …, $z_{npar} = 00101_2$. Естественно, мы должны перевести эти числа в десятичную систему счисления: $z_1 = 3$, $z_2 = 13$, …, $z_{npar} = 5$. Это означает, что приведенная на рисунке хромосома отражает следующее: параметр χ_1 является 4-м элементом массива A_1 (4-м а не 3-м, т.к. ранее мы договорились, что элементы массивов мы будем нумеровать, начиная с 1, а значит число 0_2 кодирует 1-й элемент вектора и т.д.), параметр χ_2 является 14-м элементом вектора A_2, параметр χ_{npar} является 6-м элементом вектора A_{npar}.

В данной конкретной задаче использование бинарного кодирования для представления генов имеет ряд преимуществ по сравнению с вещественным кодированием [29].

– Эволюционный алгоритм с бинарным кодированием значительно проще алгоритма с вещественным кодированием. Например, в работе [38] приведены девять возможных схем кроссовера для алгоритма с вещественным кодированием (плоский, простейший, арифметический, геометрический, дискретный,

смешанный, линейный, расширенный линейный и кроссовер, имитирующий двоичный). Таким образом, даже подбор типа скрещивания для такого алгоритма является нетривиальной задачей. В отличие от бинарного алгоритма, где надо определиться лишь с количеством точек скрещивания.

– Бинарное кодирование позволяет совмещать в одной задаче настройку как целочисленных, так и вещественных параметров. Напомним, что в пределах одного алгоритма некоторые параметры являются целочисленными, остальные же представляют собой вещественные числа. Например, в муравьином алгоритме число муравьев в колонии и число итераций (если именно оно используется в качестве условия останова) могут принимать только целые значения. А коэффициент исследования, относительная значимость феромонного уровня, относительная значимость эвристической функции, скорость испарения следа и коэффициент затухания феромона в общем случае являются вещественными числами. Таким образом, если бы мы использовали вещественное кодирование, то для некоторых генов хромосомы, кодирующей полный набор параметров, при скрещивании мы должны были бы учитывать факт целочисленности параметра (например, при помощи округления), а для других генов в этом бы не было необходимости. Данный факт значительно снижает универсальность использования вещественного кодирования, ведь для разных алгоритмов хромосома будет состоять из разного числа генов, к тому же для каждого алгоритма количество целочисленных параметров разное. Для предлагаемого нами бинарного кодирования не имеет значения, целым или вещественным числом выражается параметр. Перед началом настройки для каждого из параметров мы просто должны задать множество возможных значений этого параметра.

– Бинарное кодирование позволяет осуществлять настройку символьных (нечисловых) параметров метаэвристических алгоритмов. Под символьными параметрами подразумеваются параметры алгоритма, которые не являются числами, а «словесно» описывают некоторые его свойства. Например, в разделе

2.7 было сказано, что существует 2 разновидности пчелиного алгоритма: с использованием глобальной памяти (элитаризма) и без нее. Согласно предлагаемому в данной диссертации подходу, параметр «разновидность пчелиного алгоритма с точки зрения использования элитаризма» мы можем закодировать при помощи гена, состоящего из одного бита, где 0 будет соответствовать использованию элитаризма, а 1 – не использованию (или наоборот, это абсолютно неважно). В качестве примера других нечисловых параметров метаэвристических алгоритмов можно привести «вид селекции в эволюционном алгоритме (рулеточная, ранговая, турнирная, панмиксия)», «разновидность муравьиного алгоритма (AS, MMAS или ACS)» и т.д. Отметим, что некоторые исследователи используют другую терминологию: числовые параметры называются количественными, а нечисловые – качественными [105].

У бинарного кодирования, предлагаемого для решения задачи оптимизации управляющих параметров, есть один недостаток. Если мощность некоторого множества A_i, содержащего значения некоторого параметра, не кратна степени двойки, то встает вопрос о том, как из бинарного гена z_i получить десятичное число, отражающее позицию потенциального параметра в множестве A_i. Например, $|A_i| = 9$. Тогда для бинарного представления гена нам необходимо 4 бита, которые в действительности могут кодировать элементы множеств мощностью до 16. Представим, что мы случайно сгенерировали ген $z_i = 1011_2 = 11$. Но в нашем массиве A_i нет элемента с индексом 12. Выход из подобной ситуации прост: ген z_i кодирует индекс ind_i в массиве согласно формуле:

$$ind_i = z_i \mod L_i, \tag{3.3}$$

где L_i – число битов в гене z_i, mod – операция получения остатка от деления целых числе. Очевидно, что если величина $|A_i|$ кратна степени двойки, то $ind_i = z_i$.

Как было отмечено в разделе 2.2, эволюционные метаэвристики являются весьма молодым подходом к решению задач – ему не более 40 лет. Соответственно идеи применения эволюционного подхода к параметрической

идентификации еще моложе. Среди наиболее значимых работ, посвященных вопросам идентификации параметров систем различной сложности, отметим труды Л.А. Демидовой [12], C.L. Huang [121], W.D. Chang [87], J. Nowakova [147]. Исследования таких ученых, как A. Cachon [85], F. Ahmad [73] направлены на применение генетического алгоритма для оптимизации параметров искусственных нейронных сетей.

Вопросам настройки параметров метаэвристических алгоритмов при помощи эволюционных метаэвристик посвящены работы D. Gaertner [108], M.H. Wun [173]. В обеих работах применяется вещественное кодирование. Отметим, что в данных работах не приводится собственно алгоритм настройки параметров, в них просто на словах обозначен сам подход: применение генетического алгоритма для оптимизации параметров муравьиного [108] и пчелиного [173] алгоритма. В работе А.П. Карпенко [22] также рассматривается проблема метаоптимизации, однако не рассматривается метаметаэвристический подход как способ решения данной задачи. Таким образом, данное диссертационное исследование предлагает новый численный метод, основанный на эволюционном подходе, который позволял бы осуществлять оптимизацию управляющих параметров для любого метаэвристического алгоритма, направленного на решение задачи размещения элементов информационных систем при их планировании и оптимизации.

Отметим интересный момент. В данной работе метаэвристический алгоритм применяется для настройки управляющих параметров других метаэвристических алгоритмов. Встает вопрос о настройке управляющих параметров уже непосредственно для эволюционного алгоритма настройки параметров. Необходимо определиться с размером популяции, методом селекции, вероятностью мутации и т.д. Теоретически возможна следующая бесконечная рекурсия:

1. Разработан алгоритм для настройки оптимальных параметров метаэвристик, решающих ЗРЭРИС.

2. Необходимо придумать алгоритм для настройки управляющих параметров для эволюционного алгоритма из пункта 1.

3. Необходимо придумать алгоритм для настройки параметров для алгоритма из пункта 2.

…

И так далее. Очевидно, что на каком-то из вышеприведенных шагов необходимо прекратить оптимизацию параметров и жестко задать фиксированные значения параметров алгоритма. Лучше всего это сделать сразу на 2-м шаге. В данной диссертации для настройки оптимальных значений управляющих параметров метаэвристических алгоритмов решения ЗРЭРИС предлагается использовать эволюционный алгоритм со следующими параметрами [29]:

– число особей в популяции = 10;

– число потомков = 10;

– турнирная селекция (2 участника в каждом турнире);

– формирование новой популяции: создается промежуточная популяция из родителей и потомков;

– число точек скрещивания = 2;

– вероятность мутации = 0.05.

3.3 Псевдокод метода настройки управляющих параметров

Ниже представлена общая схема метода настройки управляющих параметров для некоторого метаэвристического алгоритма решения ЗРЭРИС, основанная на эволюционном алгоритме с бинарным кодированием [29].

1. Сформулируем тестовую задачу, на базе решения которой будет осуществляться настройка параметров.

2. Введем обозначения: *n_run* – число запусков алгоритма нахождения решения ЗРЭРИС для определения приспособленности одной особи из метода оптимизации свободных параметров; *time_alg* – время, отводимое на 1 запуск

алгоритма нахождения решения ЗРЭРИС; *time_overall* – время, отводимое на работу метода оптимизации свободных параметров.

3. Инициализируем параметры эволюционного алгоритма настройки управляющих параметров (размер популяции, вероятность мутации и т.д.).

4. Составим список управляющих параметров: зададим *npar*, множества $A_1, A_2, \ldots, A_{npar}$ (множества могут быть вещественными, целыми или нечисловыми).

5. Для каждого целого *i* от 1 до *npar*: ген, кодирующий χ_i, должен состоять из w_i битов, где w_i – минимальное из чисел, для которого выполняется $|A_i| \le 2^{w_i}$.

6. Хромосома, представляющая собой особь, состоит из *W* бит ($W = \sum_{i=1}^{npar} w_i$).

7. Формирование 1-го поколения. Сформируем N_{pop} особей следующим образом: каждая особь – двоичный вектор, каждый бит – 0 или 1 (генерируется случайно).

8. Оценка приспособленности особей популяции: запуск алгоритма решения ЗРЭРИС *n_run* раз по *time_alg* секунд и усреднение целевой функции результата.

9. Выберем N_{child} пар родителей при помощи турнирной селекции.

10. Кроссовер. В конце данного шага у нас должно быть N_{child} особей-потомков.

11. Применим оператор мутации к потомкам. Результатом мутации в ЭА с бинарным кодированием является логическая инверсия бита.

12. Формирование нового поколения. Осуществляется посредством выбора N_{pop} лучших особей из популяции, состоящей из N_{pop} родителей и N_{child} потомков.

13. Если текущее время работы меньше *time_overall*, то переход к шагу 8.

14. В качестве окончательного решения вернем набор параметров, представленный лучшей особью в популяции.

В приложении 2 (рисунок П2.4 решения ЗРЭРИС) приведена блок-схема метода настройки свободных параметров.

Особенностью предлагаемого метода является шаг №8 – оценка приспособленности особей популяции. Дело в том, что при решении ЗРЭРИС

оценка приспособленности осуществлялась очень просто: мы должны были вычислить значение согласно формуле (1.16). Приспособленность особи была обратно пропорциональна (т.к. наша задача минимизационная) целевой функции решения ЗРЭРИС.

В методе оптимизации параметров приспособленность особи вычисляется следующим образом. Каждая особь состоит из одной хромосомы, кодирующей полный набор управляющих параметров алгоритма решения ЗРЭРИС. Для каждой хромосомы мы должны *n_run* раз (по *time_alg* секунд на каждый запуск) запустить алгоритм решения ЗРЭРИС. Мы получим *n_run* значений целевой функции *F_base*. Обозначим их F_base_i, где $i \in \{1, 2, \ldots, n\_run\}$. Обозначим целевую функцию для особи в методе нахождения управляющих параметров как *Target_func*. Она вычисляется по формуле:

$$Target\_func = \frac{1}{n\_run} \cdot \sum_{i=1}^{n\_run} F\_base_i \qquad (3.4)$$

Приспособленность особи обратно пропорциональна значению *Target_func*. *Target_func* лучшей особей в последнем поколении популяции и будет равна *F_meta*.

Отметим, что предлагаемый в данной диссертации метод наиболее эффективен в ситуациях, когда число управляющих параметров (а также число возможных значений каждого из параметров) велико. Очевидно, что когда число параметров мало, то наиболее простой и эффективной стратегией будет перебор всех возможных комбинаций управляющих параметров.

Например, для пчелиного алгоритма ABC возможно 256 различных наборов управляющих параметров. Теоретически возможно решить задачу настройки параметров методом полного перебора. Однако данный подход не универсален. Например, для муравьиного алгоритма возможно примерно 2×10^{10} различных наборов управляющих параметров.

При применении эволюционного подхода к настройке параметров проверяется $N_{pop} \times N_{gen}$ возможных наборов управляющих параметров. При этом

важно отметить, что данный подход в среднем будет давать лучший результат, чем частичный перебор (нахождение лучшего набора управляющих параметров среди случайных $N_{pop} \times N_{gen}$ наборов из $|A_1| \times |A_2| \times \ldots \times |A_{npar}|$). Сама суть эволюционного алгоритма заключается в том, что особи из поколения в поколение демонстрируют всё большую приспособленность, при этом сохраняя свойства своих родителей. Таким образом, при ограниченном машинном времени эволюционный алгоритм в среднем лучше частичного перебора (экспериментальное подтверждение этого тезиса приведено в разделе 4.2.3).

3.4 Перечень оптимизируемых параметров метаэвристик

Для эволюционного алгоритма (хромосома схематично приведена на рисунке 3.6):

– $npar = 6$.

– Метод селекции. A_1 = {"рулеточная", "ранговая", "турнирная", "панмиксия"}. $|A_1| = 4$, ген кодируется 2 битами.

– Число особей в популяции (N_{pop}). $A_2 = \{5, 10, 20, 30, 40, 50, 60, 70, 80, 85, 90, 95, 100, 200, 300, 500\}$. $|A_2| = 16$, ген кодируется 4 битами.

– Число производимых за одно поколение потомков (N_{child}). $A_3 = \{N_{pop}, N_{pop} \times 2, N_{pop} \times 3, N_{pop} \times 4\}$. $|A_3| = 4$, ген кодируется 2 битами.

– Число точек скрещивания (*n_points*). $A_4 = \{1, 2, 3, 4\}$. $|A_4| = 4$, ген кодируется 2 битами.

– Вероятность мутации (p_{mut}). $A_5 = \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08\}$. $|A_5| = 8$, ген кодируется 3 битами.

– Способ формирования новой популяции. A_6 = {"создается промежуточная популяция из N_{pop} родителей и N_{child} потомков с переносом в следующее поколение N_{pop} лучших особей", "в новое поколение переходят только особи-потомки"}. $|A_6| = 2$, ген кодируется 1 битом.

Для алгоритма имитации отжига (рисунок 3.7):

– $npar = 3$.

– Коэффициент охлаждения (α). A_1 = {0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.90, 0.92, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99}. $|A_1| = 16$, ген кодируется 4 битами.

– Число итераций при одном значении температуры (*N_iter*). A_2 = {10, 20, 30, 40, 50, 60, 70, 80}. $|A_2| = 8$, ген кодируется 3 битами.

– Вероятность принятия худшего решения на ранних шагах алгоритма (p_0). A_3 = {0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.90, 0.92, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99}. $|A_3| = 16$, ген кодируется 4 битами.

Для алгоритма поиска с запретами (рисунок 3.8):

– *npar* = 3.

– Длина списка запретов (l). $A_1 = \{\lfloor N_{ps}/8 \rfloor, \lfloor N_{ps}/6 \rfloor, \lfloor N_{ps}/4 \rfloor, \lfloor N_{ps}/1 \rfloor, N_{ps}, N_{ps} \times N_{types}, (N_{ps}+N_{cl}) \times N_{types}, 2 \times (N_{ps}+N_{cl}) \times N_{types}\}$. $|A_1| = 8$, ген кодируется 3 битами.

– Параметр рандомизации окрестности (p). A_2 = {0.05, 0.1, 0.13, 0.16, 0.19, 0.22, 0.25, 0.28, 0.31, 0.34, 0.37, 0.4, 0.43, 0.46, 0.49, 0.52, 0.55, 0.58, 0.61, 0.64, 0.67, 0.7, 0.73, 0.76, 0.79, 0.82, 0.85, 0.88, 0.91, 0.94, 0.97, 1}. $|A_2| = 32$, ген кодируется 5 битами.

– Использование диверсификации в процессе поиска. A_3 = {"без диверсификации", "с диверсификацией"}. $|A_3| = 2$, ген кодируется 1 битом.

Для алгоритма мультистарта (рисунок 3.9):

– *npar* = 2.

– Максимальное число итераций подряд без улучшения целевой функции (*N_iter_max*). A_1 = {5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80}. $|A_1| = 16$, ген кодируется 4 битами.

– Дополнительное использование локального поиска. A_2 = {"без локального поиска", "с локальным поиском"}. $|A_2| = 2$, ген кодируется 1 битом.

– Параметр интенсификации поиска (θ). A_3 = {1.05, 1.1, 1.13, 1.16, 1.19, 1.22, 1.25, 1.28, 1.31, 1.34, 1.37, 1.4, 1.43, 1.46, 1.49, 1.52, 1.55, 1.58, 1.61, 1.64, 1.67, 1.7, 1.73, 1.76, 1.79, 1.82, 1.85, 1.88, 1.91, 1.94, 1.97, 2}. $|A_3| = 32$, ген кодируется 5 битами.

Для муравьиного алгоритма (рисунок 3.10):

– $npar$ = 10.

– Разновидность алгоритма. A_1 = {"AS", "MMAS", "ACS"}. $|A_1|$ = 3, ген кодируется 2 битами.

– Число муравьев (A). $A_2 = \{\lfloor N_{ps}/8 \rfloor, \lfloor N_{ps}/4 \rfloor, \lfloor N_{ps}/2 \rfloor, N_{ps}, N_{ps} \times (N_{types}-1), (N_{ps}+N_{cl}) \times (N_{types}-1), N_{ps} \times N_{types}, (N_{ps}+N_{cl}) \times N_{types}\}$. $|A_2| = 8$, ген кодируется 3 битами.

– Относительная значимость феромонного уровня для 1-й колонии (α_1). A_3 = {1, 2, 3, 4, 5, 6, 7, 8}. $|A_3| = 8$, ген кодируется 3 битами.

– Относительная значимость феромонного уровня для 2-й колонии (α_2). A_4 = {1, 2, 3, 4, 5, 6, 7, 8}. $|A_4| = 8$, ген кодируется 3 битами.

– Скорость испарения феромонного следа муравьев 1-й колонии (ρ_1). A_5 = {0.05, 0.1, 0.13, 0.16, 0.19, 0.22, 0.25, 0.28, 0.31, 0.34, 0.37, 0.4, 0.43, 0.46, 0.49, 0.52, 0.55, 0.58, 0.61, 0.64, 0.67, 0.7, 0.73, 0.76, 0.79, 0.82, 0.85, 0.88, 0.91, 0.94, 0.97, 0.99}. $|A_5| = 32$, ген кодируется 5 битами.

– Скорость испарения феромонного следа муравьев 2-й колонии (ρ_2). A_6 = {0.05, 0.1, 0.13, 0.16, 0.19, 0.22, 0.25, 0.28, 0.31, 0.34, 0.37, 0.4, 0.43, 0.46, 0.49, 0.52, 0.55, 0.58, 0.61, 0.64, 0.67, 0.7, 0.73, 0.76, 0.79, 0.82, 0.85, 0.88, 0.91, 0.94, 0.97, 0.99}. $|A_6| = 32$, ген кодируется 5 битами.

– Коэффициент затухания феромона для 1-й колонии (φ_1). A_7 = {0.05, 0.1, 0.13, 0.16, 0.19, 0.22, 0.25, 0.28, 0.31, 0.34, 0.37, 0.4, 0.43, 0.46, 0.49, 0.52, 0.55, 0.58, 0.61, 0.64, 0.67, 0.7, 0.73, 0.76, 0.79, 0.82, 0.85, 0.88, 0.91, 0.94, 0.97, 0.99}. $|A_7| = 32$, ген кодируется 5 битами.

– Коэффициент затухания феромона для 2-й колонии (φ_2). A_8 = {0.05, 0.1, 0.13, 0.16, 0.19, 0.22, 0.25, 0.28, 0.31, 0.34, 0.37, 0.4, 0.43, 0.46, 0.49, 0.52, 0.55, 0.58, 0.61, 0.64, 0.67, 0.7, 0.73, 0.76, 0.79, 0.82, 0.85, 0.88, 0.91, 0.94, 0.97, 0.99}. $|A_8| = 32$, ген кодируется 5 битами.

– Коэффициент исследования для 1-й колонии (q_0^1). A_9 = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}. $|A_9| = 8$, ген кодируется 3 битами.

– Коэффициент исследования для 2-й колонии (q_0^2). A_{10} = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}. $|A_{10}| = 8$, ген кодируется 3 битами.

Отметим, что мы не оптимизируем значения относительной значимости эвристической функции для 1-й и 2-й колонии муравьев, т.к. эти параметры можно зафиксировать (например, принять их равными единице) и варьировать только относительную значимость феромонного уровня [108, 168].

Для пчелиного алгоритма BCO (рисунок 3.11):

– $npar = 4$.

– Метод селекции. A_1 = {"рулеточная", "ранговая", "турнирная", "дизруптивная"}. $|A_1| = 4$, ген кодируется 2 битами.

– Число пчел (B). $A_2 = \{3, 4, 5, 6, 7, 8, 9, 10\}$. $|A_2| = 8$, ген кодируется 3 битами.

– Число прямых/обратных проходов в пределах одной итерации (NC). $A_3 = \{3, 5, 7, 10, 12, 15, 20, 25\}$. $|A_3| = 8$, ген кодируется 3 битами.

– Использование глобальной памяти. A_4 = {"без глобальной памяти", "с глобальной памятью"}. $|A_4| = 2$, ген кодируется 1 битом.

Для пчелиного алгоритма ABC (рисунок 3.12):

– $npar = 3$.

– Метод селекции. A_1 = {"рулеточная", "ранговая", "турнирная", "дизруптивная"}. $|A_1| = 4$, ген кодируется 2 битами.

– Число источников нектара (SN). $A_2 = \{10, 20, 30, 40, 50, 60, 70, 80\}$. $|A_2| = 8$, ген кодируется 3 битами.

– Максимальное число итераций подряд без улучшения целевой функции ($limit$). $A_3 = \{\lfloor SN \times N_{ps}/8 \rfloor, \lfloor SN \times N_{ps}/4 \rfloor, \lfloor SN \times N_{ps}/2 \rfloor, SN \times N_{ps}, SN \times N_{ps} \times (N_{types} - 1), SN \times (N_{ps} + N_{cl}) \times (N_{types} - 1), SN \times N_{ps} \times N_{types}, SN \times (N_{ps} + N_{cl}) \times N_{types}\}$. $|A_3| = 8$, ген кодируется 3 битами.

Отметим интересный момент. Например, требуется найти набор управляющих параметров для муравьиного алгоритма. Пусть χ_1 = "AS". Тогда нет никакой разницы, какие значения будут принимать параметры $q_0^1(\chi_9)$ и $q_0^2(\chi_{10})$,

т.к. в классическом алгоритме AS они не используются. Соответственно, при расчете функции приспособленности участки хромосомы, отвечающие за кодирование параметров χ_9 и χ_{10}, участвовать не будут. Данный аспект функционирования эволюционных алгоритмов не противоречит современным научным представлениям о генетике (а именно на аналогии с реальной природной эволюцией и стоящей за ней генетикой и базируется концепция эволюционных алгоритмов), согласно которым в структуре ДНК имеются достаточно крупные участки, никаким образом не влияющие на генотип, а, следовательно, и на фенотип (гены, не содержащие генетическую информацию – интроны) [20].

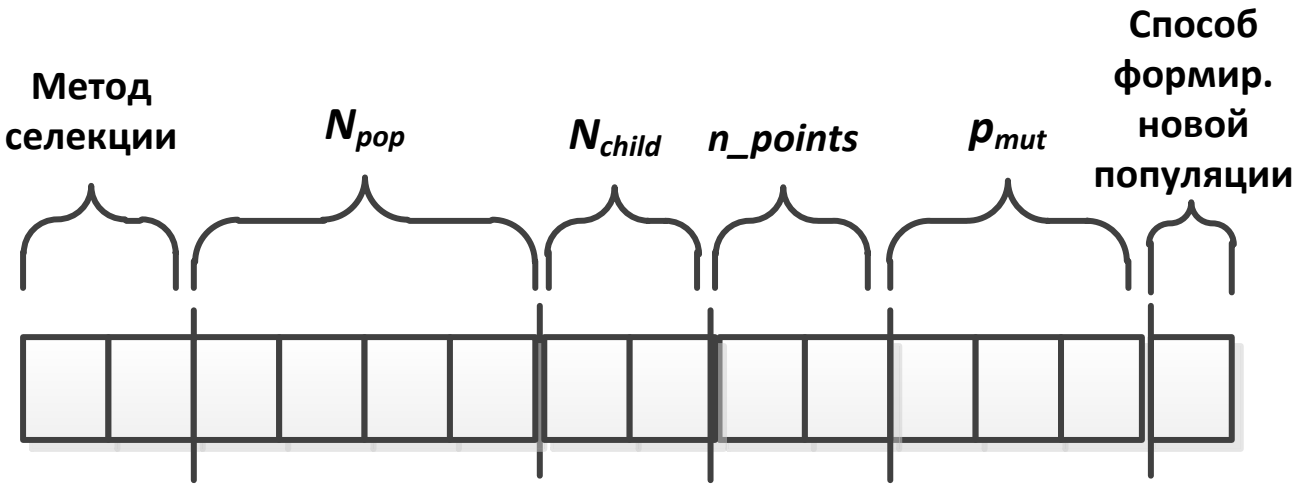

Рисунок 3.6 – Хромосома для эволюционного алгоритма

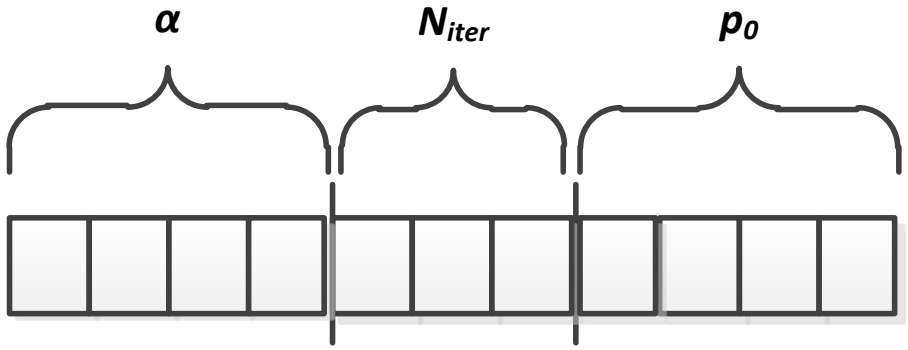

Рисунок 3.7 – Хромосома для алгоритма имитации отжига

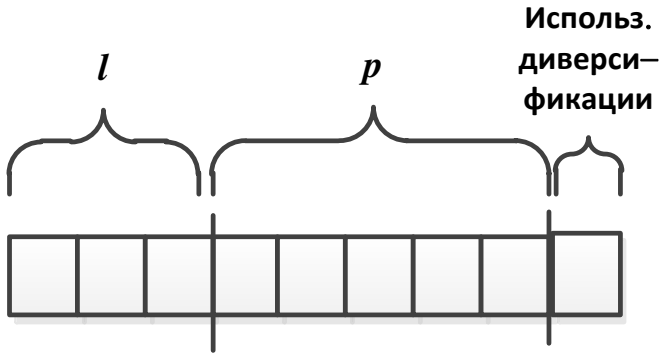

Рисунок 3.8 – Хромосома для алгоритма поиска с запретами

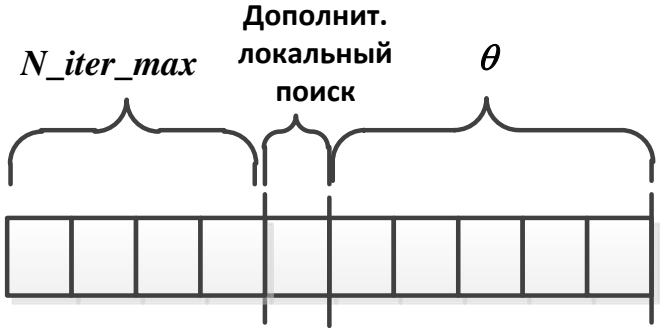

Рисунок 3.9 – Хромосома для алгоритма мультистарта

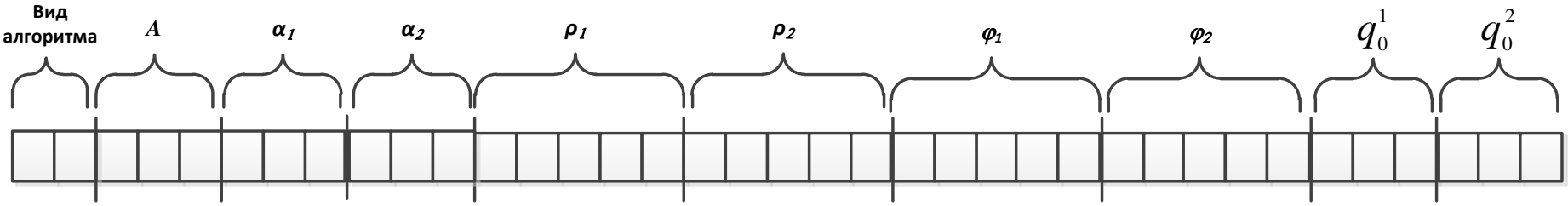

Рисунок 3.10 – Хромосома для муравьиного алгоритма

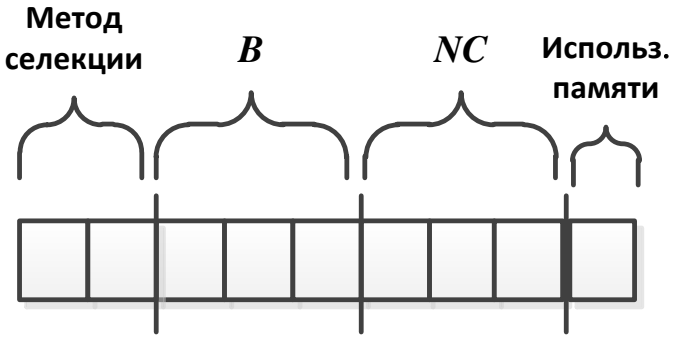

Рисунок 3.11 – Хромосома для алгоритма BCO

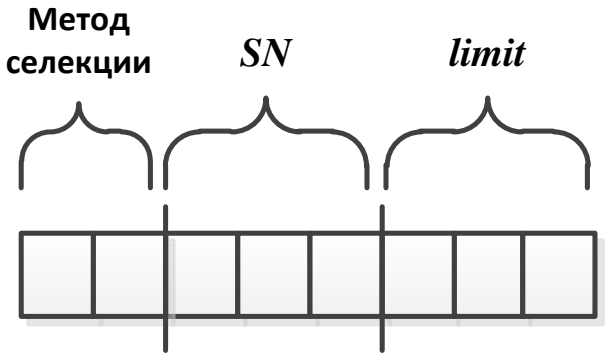

Рисунок 3.12 – Хромосома для алгоритма ABC

3.5 Значения управляющих параметров «по умолчанию»

В таблице 3.1 приведены значения «по умолчанию» для управляющих параметров метаэвристических алгоритмов, перечисленных в разделе 3.4. Под параметрами «по умолчанию» подразумеваются те значения управляющих параметров, которые рекомендуются в работах авторов классических метаэвристических алгоритмов, а также в ряде других авторитетных научных книг и статей (ссылки на работы приведены в ячейках первого столбца таблицы 3.1).

Таблица 3.1 – Значения управляющих параметров «по умолчанию» для разработанных метаэвристических алгоритмов

Алгоритм	Набор параметров
Эволюционный алгоритм [41,102,119,134]	а) метод селекции = "рулеточная"; б) N_{pop} = 100; в) $N_{child} = N_{pop}$; г) *n_points* = 1; д) p_{mut} = 0.05; е) способ формирования новой популяции = "в новое поколение переходят только особи-потомки".
Имитация отжига [127,90,170,134]	а) α = 0.99; б) *N_iter* = 10; в) p_0 = 0.9.
Поиск с запретами [112,26,110]	а) l = 100; б) p = 0.05; в) использование диверсификации в процессе поиска = "с диверсификацией".
Мультистарт [110,79,133]	а) *N_iter_max* = 100; б) дополнительное использование локального поиска = "без локального поиска"; в) θ = 1.25;.
Пчелиный алгоритм BCO [166,167]	а) метод селекции = "рулеточная"; б) B = 5; в) NC = 15; г) использование памяти = "с глобальной памятью".
Пчелиный алгоритм ABC [123,124]	а) метод селекции = "рулеточная"; б) SN = 10; в) *limit* = 100.

Продолжение таблицы 3.1

Алгоритм	Набор параметров
Муравьиный алгоритм [100,97,162,168]	а) разновидность алгоритма = "ACS"; б) $A = N_{ps}$; в) $\alpha_1 = 3$; г) $\alpha_2 = 3$; д) $\rho_1 = 0.1$; е) $\rho_2 = 0.1$; ж) $\varphi_1 = 0.1$; з) $\varphi_2 = 0.1$; и) $q_0^1 = 0.5$; к) $q_0^2 = 0.5$.

3.6 Вторая фаза настройки оптимальных параметров метаэвристических алгоритмов

Назовем эволюционный алгоритм с бинарным кодированием, представленный выше, первой фазой метода оптимизации значений управляющих параметров метаэвристических алгоритмов решения задачи размещения элементов развивающихся информационных систем.

В случае наличия дополнительного машинного времени возможен запуск второй фазы поиска [52]. Вторая фаза представляет собой ЭА с вещественным кодированием. Представление набора параметров отличается от 1-й фазы. Хромосома все так же состоит из *npar* генов. Фенотип представляет собой набор из *npar* переменных управляющих параметров: $\{\chi_1, \chi_2, \ldots, \chi_{npar}\}$. Только каждый ген уже не кодируется совокупностью бит, а в явном виде представляет собой вещественное число, является целочисленным или нечисловым (рисунок 3.13). Таким образом, в ЭА с вещественным кодированием фенотип совпадает с генотипом, а поиск фактически производится в пространстве фенотипов [38].

Пусть по итогам 1-й фазы поиска у нас есть решение-хромосома. В процессе 2-й фазы те гены, которые являются целочисленным и символьными, меняться не будут. Будет производиться дальнейшая оптимизация только вещественных параметров (т.е. меняться будут только гены, выраженные вещественными

числами). Таким образом, для пчелиных алгоритмов BCO и ABC 2-я фаза не нужна, т.к. у них нет управляющих параметров, являющихся вещественными числами.

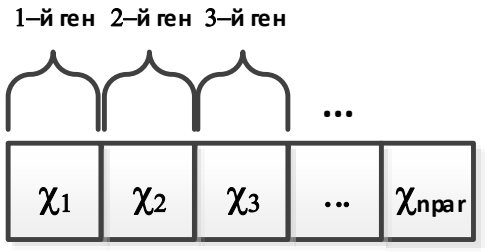

Рисунок 3.13 – Вторая фаза настройки управляющих параметров. Представление параметров алгоритма в виде хромосомы

Ниже представлена общая схема 2-й фазы поиска для метода оптимизации управляющих параметров некоторого метаэвристического алгоритма решения задачи размещения элементов информационных систем при их планировании и оптимизации.

1. Введем обозначения: *n_run* – число запусков алгоритма нахождения решения ЗРЭРИС для определения приспособленности одной особи из метода настройки параметров; *time_alg* – время, отводимое на 1 запуск алгоритма нахождения решения ЗРЭРИС; *time_overall* – время, отводимое на работу 2-й фазы метода настройки параметров.

2. Инициализация параметров ЭА. Размер популяции, число потомков, вероятность мутации и тип селекции такие же, как и в 1-й фазе.

3. Формирование 1-го поколения. По итогам 1-й фазы у нас есть некоторое решение. Это набор параметров $\{\chi_1^*, \chi_2^*, \ldots, \chi_{npar}^*\}$. Первая хромосома исходной популяции совпадает с лучшим решением 1-й фазы. Для остальных хромосом 2-й фазы значение каждого вещественного гена χ_i сгенерируем по формуле

$$\chi_i = \left[\chi_i^* + j \cdot (\chi_i^{\max} - \chi_i^{\min})\right]_{\chi_i^{\min}}^{\chi_i^{\max}}, \qquad (3.5)$$

где j – случайное вещественное число из диапазона (–0.05; 0.05), $\chi_i^{\min}$ – минимально допустимое значение i-го параметра, $\chi_i^{\max}$ – максимально допустимое значение i-го параметра.

4. Оценка приспособленности особей популяции.

5. Выбор N_{child} пар родительских особей при помощи турнирной селекции.

6. Кроссовер. В конце данного шага у нас должно быть N_{child} особей-потомков. Используется плоский кроссовер [38].

7. Применение оператора мутации к потомкам. Используется случайная мутация [38].

8. Формирование нового поколения. Осуществляется посредством выбора N_{pop} лучших особей из популяции, состоящей из N_{pop} родителей и N_{child} потомков.

9. Если текущее время работы меньше *time_overall*, то переход к шагу 4.

10. В качестве окончательного решения вернем набор параметров, представленный лучшей особью в популяции.

3.7 Выводы

1. Сформулирована задача оптимизации параметров метаэвристических алгоритмов. Предлагается решать данную задачу при помощи метаэвристики более высокого уровня. Данный подход носит название метаметаэвристического и лежит в основе процесса метаоптимизации.

2. Предложен метод оптимизации параметров метаэвристических алгоритмов решения задачи размещения элементов развивающихся информационных систем, представленных в главе 2, основанный на эволюционном подходе. Предложенный метод подразумевает две фазы поиска оптимального решения: эволюционный алгоритм с бинарным кодированием и эволюционный алгоритм с вещественным кодированием.

4 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СОЗДАННЫХ МЕТОДОВ И АЛГОРИТМОВ И ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

4.1 Программная реализация предложенных методов и алгоритмов

Программный комплекс, созданный в рамках диссертационного исследования, имеет структуру, приведенную на рисунке 4.1.

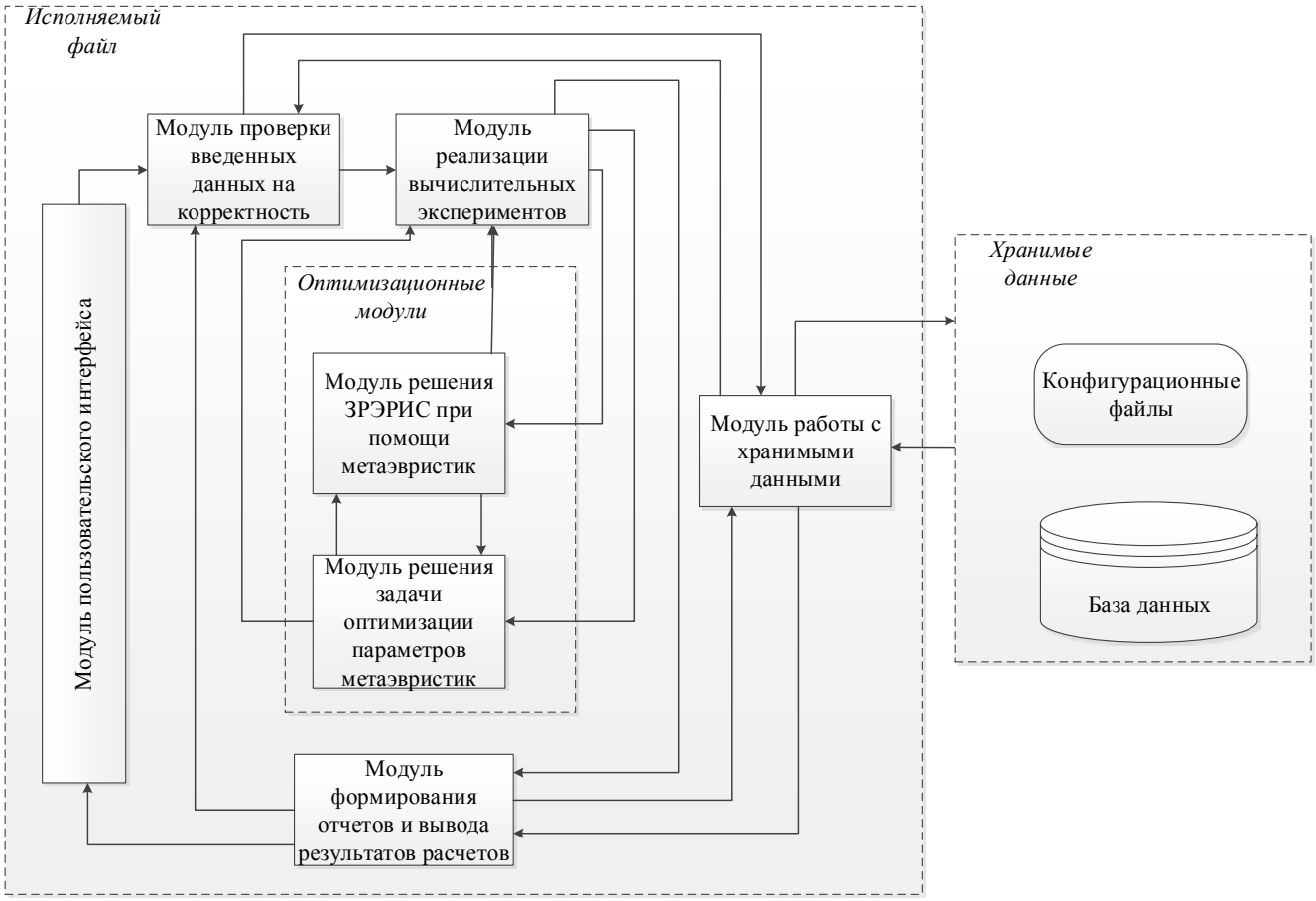

Рисунок 4.1 – Структура созданного программного комплекса

Для реализации представленных в главах 2 и 3 алгоритмов и методов был выбран язык программирования Object Pascal. Программный комплекс интеллектуальной поддержки принятия решений, реализующий алгоритмы размещения элементов развивающихся информационных систем, был создан в среде Embarcadero Delphi 2010. Основу программного кода составили алгоритмы, псевдокод и блок-схемы которых приведены в главах 2 и 3 и в тексте приложений. Разработанное ПО направлено на оптимизацию размещения

элементов одной из разновидностей современных развивающихся информационных систем – беспроводных сетей передачи данных. Особенностью разработанного ПО является учет в математической модели раздела 1.4 особенностей беспроводных сетей, описанных в разделе 1.5.

Созданный программный комплекс интеллектуальной поддержки принятия решений подразумевает работу с двумя типами хранимых данных: конфигурационными файлами и данными, хранящимися в реляционной базе данных. Конфигурационные файлы представляют собой обычные текстовые файлы. База данных выполнена на основе СУБД MySQL 5.5. Особенностью предлагаемого подхода к организации хранимых данных является тот факт, что некоторые данные, например, параметры среды распространения беспроводного сигнала, могут храниться как в конфигурационных файлах, так и в базе данных. Т.е. в принципе, возможна работа без наличия соединения с базой данных, только на основе данных, хранящихся в конфигурационных файлах. Но в таком режиме пользователю недоступны результаты предыдущих расчетов и серий вычислительных экспериментов, которые могут быть необходимы для корректного сравнения разных алгоритмов между собой.

Схематично перечень хранимых данных, используемых в созданном программном комплексе, представлен на рисунке 4.2.

Отличительной особенностью структуры созданного программного комплекса является наличие инструментов автоматизации проведения серий вычислительных экспериментов с целью оценки качества алгоритмов, лежащих в основе созданной программной системы. Дело в том, что для проведения корректного сравнения между собой различных алгоритмов (или разных модификаций одного алгоритма) необходимо многократно запустить функционирование этого алгоритма, при этом «запомнив» результаты расчетов после каждого запуска. Очевидно, что делать это вручную крайне затруднительно. Модуль реализации вычислительных экспериментов, приведенный на рисунке 4.1, предназначен именно для этой цели. Результаты

работы именно этого модуля легли в основу раздела 4.2. Пример окна запуска серии вычислительных экспериментов приведен на рисунке 4.3.

Рисунок 4.2 – Хранимые данные

Представленный в данной главе программный комплекс имеет в своей основе программы, зарегистрированные в Федеральной службе по интеллектуальной собственности РФ («Эволюционный алгоритм решения задачи размещения базовых станций» [62], «Программа для решения задачи размещения базовых станций методами поиска с запретами и мульти-старта» [59], «Программа для решения задачи планирования беспроводных сетей передачи данных при помощи метаэвристических методов оптимизации» [58]). Скан-копии свидетельств о регистрации ПО приведены в Приложении 5.

Отдельные модули данного программного комплекса интеллектуальной поддержки принятия решений были внедрены в ОАО "Мобильные ТелеСистемы" (филиал в Липецкой области) и АО "ЭР-Телеком Холдинг" (филиал в г. Липецк). Скан-копии актов о внедрении приведены в Приложении 4.

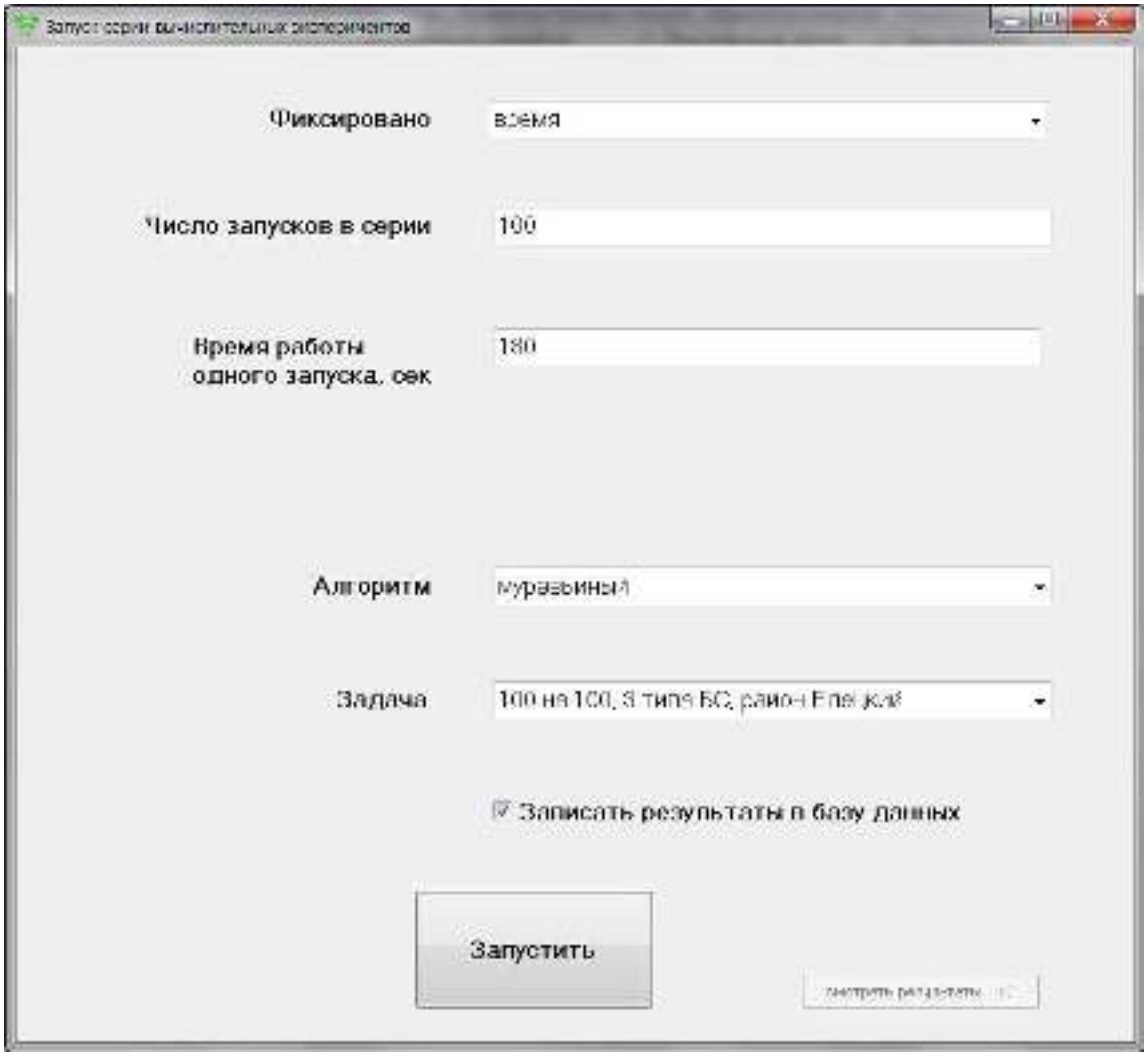

Рисунок 4.3 – Вид окна запуска серий вычислительных экспериментов

В приложении 3 приведены скриншоты некоторых окон графического интерфейса программного комплекса и кратко описан функционал ПО.

4.2 Вычислительный эксперимент на базе созданного программного обеспечения

Компьютерные эксперименты проводились на компьютере с процессором Intel Core i5-3470 и оперативной памятью 6 ГБ. Для проведения вычислительных экспериментов предварительно было сгенерировано множество тестовых задач различной размерности.

Для сравнения между собой разных алгоритмов (или разных вариаций одного алгоритма) были выбраны три критерия качества алгоритмов:

1) Оценка относительной погрешности решений (δ). Для конкретного алгоритма она рассчитывается следующим образом. Пусть F_{best} – целевая функция точного (оптимального) решения задачи. Но для задач средней и большой размерности, как правило, не представляется возможным (из-за необходимости крайне долго искать решение при помощи полного перебора) найти точное решение. Поэтому для нахождения δ используется F_{best_known} – целевая функция лучшего из известных решений задачи (т.н. рекорда). Относительная погрешность некоторого i-го запуска алгоритма:

$$\delta_i = \frac{|F_{best} - F_i|}{F_{best}} \approx \frac{|F_{best\_known} - F_i|}{F_{best\_known}}. \quad (4.1)$$

Пусть каждый из алгоритмов с конкретным набором значений управляющих параметров запускается N_{run} раз. Тогда среднее значение относительной погрешности решений конкретного алгоритма:

$$\bar{\delta} = \sum_{j=1}^{N_{run}} \delta_j / N_{run}. \quad (4.2)$$

Для корректного сравнения для всех алгоритмов фиксируется одно и то же время работы алгоритма.

2) Оценка среднего времени сходимости (T) – время, измеряемое от запуска алгоритма до момента достижения рекорда с заданной точностью ε, усредненное по статистически независимым запускам.

3) Оценка доли успешных запусков (*succ*) – отношение числа успешных запусков, в которых алгоритм оптимизации обнаруживает решение заранее заданного качества, к общему числу статистически независимых запусков, выраженное в процентах [11].

4.2.1 Сравнение разработанных алгоритмов с методом полного перебора

Первая серия вычислительных экспериментов была посвящена исследованию быстродействия и точности предложенных в главе 2 алгоритмов путем сравнения их с методом полного перебора (ПП). Т.к. метод ПП не

позволяет решать задачу за полиномиальное время, сравнение проводилось на задачах малой и средней размерности.

Зафиксируем параметры метаэвристических алгоритмов. Эволюционный алгоритм: $N_{pop} = 40$, $N_{child} = N_{pop} \times 2$, $p_{mut} = 0.03345$, $n\_points = 3$, турнирная селекция, популяции потомков и родителей сливаются. Алгоритм имитации отжига: $p_0 = 0.95712$, $\alpha = 0.97854$, $N_{iter} = 60$. Алгоритм поиска с запретами: $l = N_{ps}$, $p = 0.15615$, используется диверсификация. Алгоритм мультистарта: $N\_iter\_max = 50$, $\theta = 0.31773$, без дополнительного использование локального спуска. Муравьиный алгоритм: ACS, $A = N_{ps}$, $\alpha_1 = 3$, $\alpha_2 = 1$, $\rho_1 = 0.09574$, $\rho_2 = 0.11055$, $\varphi_1 = 0.12560$, $\varphi_2 = 0.10042$, $q_0^1 = 0.14061$, $q_0^2 = 0.22816$. Пчелиный алгоритм BCO: $B = 5$, $NC = 12$, use_global = true, метод селекции = "турнирная". Пчелиный алгоритм ABC: $SN = 20$, $limit = SN \times N_{ps}$, метод селекции = "турнирная". Для всех алгоритмов условием останова будет превышение времени функционирования $time = 0.1$ с.

Результаты работы алгоритмов приведены в таблице 4.1. Под размером задачи подразумевается $N_{cl} \times N_{ps} \times N_{types}$. Каждая ячейка таблицы 4.1 содержит две строки: верхняя – это время работы алгоритма в секундах при решении задачи определенной размерности, нижняя – соответствующее значение оценки относительной погрешности решений. Время решения задачи и погрешность для всех алгоритмов кроме полного перебора приводятся как среднее за 100 запусков алгоритма. Т.к. алгоритм полного перебора является точным, достаточно всего одного запуска. Очевидно, что для ПП значение погрешности равно нулю.

Представленные в таблице 4.1 данные свидетельствуют о том, что на задачах малой размерности предложенные алгоритмы обеспечивают получение точных значений целевой функции практически при каждом запуске алгоритма. При этом предложенные алгоритмы справляются с решением ЗРЭРИС (на примере беспроводных сетей передачи данных) за очень малое время, в отличие от полного перебора.

Таблица 4.1 – Сравнение созданных алгоритмов и метода полного перебора

Размерность задачи	Алгоритм							
	ПП	ЭА	ИО	ПЗ	МС	АСО	ВСО	АВС
3×5×2	0.030 с 0%	0.102 с 0 %	0.102 с 0 %	0.104 с 0 %	0.101 с 0 %	0.103 с 0 %	0.101 с 0 %	0.101 с 0 %
3×7×2	0.710 с 0%	0.103 с 0 %	0.103 с 0 %	0.101 с 0 %	0.102 с 0 %	0.101 с 0 %	0.102 с 0 %	0.101 с 0 %
3×10×2	65.19 с 0%	0.104 с 0 %	0.103 с 0 %	0.102 с 0 %	0.103 с 0 %	0.104 с 0 %	0.105 с 0 %	0.102 с 0 %
5×5×2	0.606 с 0%	0.102 с 0 %	0.101 с 0 %	0.101 с 0 %	0.102 с 0 %	0.103 с 0 %	0.104 с 0 %	0.101 с 0 %
5×7×2	31.59 с 0%	0.103 с 0 %	0.101 с 0.003 %	0.103 с 0 %	0.102 с 0 %	0.101 с 0 %	0.103 с 0 %	0.101 с 0 %
5×10×2	5141 с 0%	0.101 с 0.003 %	0.102 с 0 %	0.103 с 0 %	0.101 с 0.009%	0.103 с 0.003 %	0.102 с 0 %	0.103 с 0.011%
7×5×2	14.65 с 0%	0.103 с 0 %	0.103 с 0 %	0.103 с 0 %	0.101 с 0 %	0.101 с 0 %	0.101 с 0 %	0.102 с 0 %
7×7×2	1390 с 0%	0.102 с 0.002 %	0.102 с 0 %	0.102 с 0.007 %	0.102 с 0 %	0.102 с 0 %	0.101 с 0.011%	0.101 с 0 %
7×10×2	455798 с 0%	0.105 с 0.004 %	0.101 с 0.012 %	0.101 с 0.008 %	0.104 с 0.012 %	0.103 с 0 %	0.101 с 0.009 %	0.101 с 0.007 %

4.2.2 Сравнение имитации отжига, поиска с запретами и мультистарта с алгоритмом локального спуска

Также был проведен ряд вычислительных экспериментов с целью сравнить эффективность алгоритмов локального спуска, ИО, ПЗ и МС при решении задач разной размерности. Такое сравнение необходимо, т.к. последние три алгоритма являются алгоритмами, базирующимися на локальном поиске, и в теории должны давать результат лучше, чем базовый алгоритм локального спуска.

Зафиксируем параметры метаэвристик. Алгоритм имитации отжига: $p_0 = 0.95712$, $\alpha = 0.97854$, $N_{iter} = 60$. Алгоритм поиска с запретами: $l = N_{ps}$, $p = 0.15615$, используется диверсификация. Алгоритм мультистарта: $N\_iter\_max = 50$, $\theta = 0.31773$, без дополнительного использования локального спуска. Моделирование производилось следующим образом: сначала каждую из задач мы решали при помощи алгоритма локального спуска (усредняя результаты за 100 запусков). Далее мы решали ту же задачу алгоритмами ИО, ПЗ и МС (те же 100 запусков для каждого алгоритма), для корректности сравнения ограничивая их работу временем, которое понадобилось методу локального спуска для решения задачи. Результаты эксперимента приведены в таблице 4.2. Каждая ячейка таблицы содержит две строки: верхняя – это время работы алгоритма в секундах, нижняя – значение оценки относительной погрешности решений, получаемых данным алгоритмом. Из таблицы видно, что алгоритмы ИО, ПЗ и МС решают ЗРЭРИС лучше, чем алгоритм локального спуска.

Таблица 4.2 – Сравнение результатов локального поиска, имитации отжига, поиска с запретами и мультистарта

Размерность задачи	Алгоритм			
	локальный спуск	ИО	ПЗ	МС
50×50×3	0.441 с 32.775 %	0.445 с 21.674 %	0.446 с 18,598 %	0.445 с 12,828 %
100×100×3	1.191 с 32.775 %	1.197 с 19.435 %	1.193 с 15,194 %	1.196 с 13,170 %

Продолжение таблицы 4.2

Размерность задачи	Алгоритм			
	локальный спуск	ИО	ПЗ	МС
150×150×3	2,535 с 31,690 %	2,541 с 19.315 %	2,539 с 17,039 %	2,538 с 13,299 %
200×200×3	5,933 с 38,305 %	5,939 с 22.097 %	5,942 с 16,130 %	5,936 с 13,700 %
250×250×3	9,389 с 34,008 %	9,392 с 17.054 %	9,399 с 15.489 %	9,396 с 12.984 %
300×300×3	14.540 с 29.995 %	14.546 с 16.436 %	14.545 с 14.547 %	14.542 с 12.236 %
350×350×3	19.675 с 40.091 %	19.678 с 15.023 %	19.678 с 13.562 %	19.678 с 11.947 %
400×400×3	28.220 с 36.698 %	28.226 с 14.019 %	28.223 с 11.489 %	28.223 с 10.536 %
450×450×3	41.166 с 31.022 %	41.170 с 11.162 %	41.168 с 9.488 %	41.169 с 8.923 %
500×500×3	57.056 с 38.514%	57.059 с 9.355 %	57.058 с 9.664 %	57.061 с 7.561 %

4.2.3 Настройка оптимальных параметров алгоритмов

Следующая серия вычислительных экспериментов была посвящена сравнению эффективности метода, предлагаемого в главе 3, и метода частичного перебора при нахождении наилучшего набора управляющих параметров. В качестве тестового примера была выбрана задача следующей размерности: $N_{cl} = 100$, $N_{ps} = 100$, $N_{types} = 3$. Зафиксируем параметры предлагаемого эволюционного алгоритма нахождения параметров метаэвристик: $N_{pop} = 10$, $N_{child} = 10$, $N_{gen} = 19$, $n\_run = 6$, $time\_alg = 10$ с. Таким образом, в процессе работы алгоритма мы рассмотрим 200 наборов управляющих параметров для алгоритмов из главы 2. Соответственно для корректного сравнения в методе перебора мы должны рассмотреть 200 случайных наборов управляющих параметров.

Результаты эксперимента приведены в таблице 4.3. Каждая ячейка таблицы 4.3 содержит две строки: верхняя – это время работы алгоритма настройки в

минутах, нижняя – соответствующее усредненное значение оценки относительной погрешности решений для 10 лучших наборов управляющих параметров (очевидно, что лучший набор управляющих параметров – тот, который дает наименьшее значение целевой функции). Из таблицы 4.3 видно, что предлагаемый в главе 3 метод настройки управляющих параметров для метаэвристик решения ЗРЭРИС в среднем позволяет достичь лучших результатов, чем метод частичного перебора.

Таблица 4.3 – Сравнение предлагаемого эволюционного метода и перебора

Настраиваемый алгоритм	Алгоритм настройки параметров	
	Эволюционный	Перебор
Эволюционный алгоритм	202.44 мин 9.668 %	200.81 мин 11.543 %
Имитация отжига	202.67 мин 9.368 %	200.58 мин 10.823 %
Поиск с запретами	202.15 мин 9.260 %	200.03 мин 10.449 %
Мультистарт	202.65 мин 8.962 %	200.44 мин 10.962 %
Муравьиный алгоритм	202.49 мин 7.069 %	200.62 мин 10.072 %
Пчелиный алгоритм BCO	202.23 мин 6.264 %	200.96 мин 6.991 %
Пчелиный алгоритм ABC	202.92 мин 8.689 %	200.43 мин 9.273 %

Дальнейшие эксперименты относились к проблеме нахождения оптимальных значений параметров метаэвристических алгоритмов решения ЗРЭРИС. Перечень оптимизируемых параметров приведен в разделе 3.4. В таблице 4.4 приведены результаты настройки управляющих параметров при помощи алгоритма, предложенного в главе 3. Параметры эволюционного метода настройки параметров были следующими: $N_{pop} = 10$, $N_{child} = 10$, $n\_run = 6$, $time\_alg = 10$ с, $time\_overall = 360$ мин. На 2-ю фазу поиска было выделено по 180 минут для каждого алгоритма (кроме пчелиных, для которых 2-я фаза не предусмотрена, т.к. у них нет вещественных параметров).

Таблица 4.4 – Результаты настройки управляющих параметров для разработанных метаэвристических алгоритмов

Алгоритм	Набор параметров
Эволюционный алгоритм	а) метод селекции = "турнирная"; б) $N_{pop} = 40$; в) $N_{child} = N_{pop} \times 2$; г) $n\_points = 3$; д) $p_{mut} = 0.03345$; е) способ формирования новой популяции = "создается промежуточная популяция из N_{pop} родителей и N_{child} потомков с переносом в следующее поколение N_{pop} лучших особей".
Имитация отжига	а) $\alpha = 0.97854$; б) $N\_iter = 60$; в) $p_0 = 0.95712$.
Поиск с запретами	а) $l = N_{ps}$; б) $p = 0.15615$; в) использование диверсификации в процессе поиска = "с диверсификацией".
Мультистарт	а) $N\_iter\_max = 50$; б) дополнительное использование локального поиска = "с локальным поиском"; в) $\theta = 0.31773$;.
Муравьиный алгоритм	а) разновидность алгоритма = "ACS"; б) $A = N_{ps}$; в) $\alpha_1 = 3$; г) $\alpha_2 = 1$; д) $\rho_1 = 0.09574$; е) $\rho_2 = 0.11055$; ж) $\varphi_1 = 0.12560$; з) $\varphi_2 = 0.10042$; и) $q_0^1 = 0.14061$; к) $q_0^2 = 0.22816$.
Пчелиный алгоритм BCO	а) метод селекции = "турнирная"; б) $B = 5$; в) $NC = 12$; г) использование памяти = "с глобальной памятью".
Пчелиный алгоритм ABC	а) метод селекции = "турнирная"; б) $SN = 20$; в) $limit = SN \times N_{ps}$.

4.2.4 Сравнение разработанных алгоритмов между собой

Далее были проведены две серии вычислительных экспериментов, направленных на доказательство необходимости процедуры настройки управляющих параметров для метаэвристических алгоритмов решения оптимизационных задач на примере ЗРЭРИС и для сравнения разработанных алгоритмов между собой. Для предложенных в главе 2 алгоритмов сравнивалась эффективность набора управляющих параметров «по умолчанию» (см. таблицу 3.1) и набора параметров, найденного при помощи эволюционного метода, предложенного в главе 3 (см. таблицу 4.4).

Первая серия экспериментов производилась следующим образом: каждую из задач оптимизации структуры беспроводной сети передачи данных мы решали семью алгоритмами с двумя разными наборами параметров для каждого алгоритма, отводя на нахождение решения одинаковое время (5 минут для задачи 50×50×3, 10 мин для 100×100×3, 15 мин для 150×150×3, 20 мин для 200×200×3, 25 мин для 250×250×3, 30 мин для 300×300×3, 35 мин 350×350×3, 40 мин для 400×400×3, 45 мин для 450×450×3, 50 мин для 500×500×3) и усредняя результаты по итогам 30 запусков. Результаты эксперимента приведены в таблице 4.5. Каждая ячейка таблицы 4.5 содержит две строки: верхняя – это значение оценки относительной погрешности решений задачи определенной размерности для соответствующего алгоритма с набором управляющих параметров, найденных при помощи предлагаемого в диссертации алгоритма (см. таблицу 4.4), нижняя – аналогичная величина для управляющих параметров «по умолчанию» (см. раздел 3.5).

Вторая серия экспериментов производилась аналогично первой. С одним изменением: был установлен другой лимит времени (5 секунд для задачи 50×50×3, 10 с для 100×100×3, 15 с для 150×150×3, 20 с для 200×200×3, 25 с для 250×250×3, 30 с для 300×300×3, 35 с для 350×350×3, 40 с для 400×400×3, 45 с для 450×450×3, 50 с для 500×500×3) и результаты усреднялись по итогам 100 запусков. Результаты эксперимента приведены в таблице 4.6.

Следующая серия экспериментов была направлена на то, чтобы оценить предложенные алгоритмы по критерию оценки среднего времени сходимости. Результаты эксперимента, усредненные по 100 запускам, приведены в таблицах 4.7 (для $\varepsilon = 5\ \%$) и 4.8 (для $\varepsilon = 10\ \%$). Ячейки таблиц содержат значение T для запусков алгоритма с настроенными управляющими параметрами (нижняя строка каждой ячейки) и для запусков алгоритма с параметрами «по умолчанию» (верхняя строка каждой ячейки).

Следующая серия экспериментов была направлена на то, чтобы оценить предложенные алгоритмы по критерию оценки доли успешных запусков. Точность – 1 % от значения рекорда, время: 50 минут для задачи 50×50×3, 100 мин для 100×100×3, 150 мин для 150×150×3, 200 мин для 200×200×3, 250 мин для 250×250×3, 300 мин для 300×300×3, 350 мин 350×350×3, 400 мин для 400×400×3, 450 мин для 450×450×3, 500 мин для 500×500×3. Результаты эксперимента, усредненные по 20 запускам, приведены в таблице 4.9. Ячейки таблицы содержат значение *succ* для запусков алгоритма с настроенными управляющими параметрами (нижняя строка каждой ячейки) и для запусков алгоритма с параметрами «по умолчанию» (верхняя строка каждой ячейки).

Из таблиц 4.5–4.9 можно сделать два основных вывода:

– настройка управляющих параметров способна значительно увеличить эффективность метаэвристик по сравнению с использованием параметров «по умолчанию»;

– пчелиный алгоритм BCO лучше других алгоритмов справляется с решением задачи размещения элементов развивающхся информационных систем. (на примере беспроводной сети передачи данных).

Таблица 4.5 – Результаты сравнения разных наборов управляющих параметров для разработанных метаэвристических алгоритмов (по критерию оценки относительной погрешности решений). Серия экспериментов №1

Размерность задачи	**Алгоритм**						
	ЭА	**ИО**	**ПЗ**	**МС**	**АСО**	**ВСО**	**АВС**
50×50×3	6.216 % 7.271 %	5.525 % 6.109 %	5.070 % 5.706 %	3.021 % 4.602 %	2.531 % 4.593 %	2.636 % 3.793 %	4.513 % 5.539 %
100×100×3	5.585 % 6.558 %	5.850 % 6.467 %	5.402 % 6.364 %	4.498 % 4.987 %	3.496 % 4.483 %	2.233 % 3.292 %	3.531 % 5.569 %
150×150×3	6.104 % 6.495 %	5.542 % 6.084 %	4.413 % 5.061 %	3.109 % 3.934 %	3.507 % 3.657 %	3.073 % 4.212 %	3.639 % 4.984 %
200×200×3	6.082 % 6.648 %	4.880 % 5.831 %	4.616 % 5.703 %	3.508 % 3.798 %	2.872 % 3.846 %	2.182 % 3.651 %	3.502 % 5.255 %
250×250×3	6.072 % 7.030 %	5.211 % 6.149 %	5.246 % 5.835 %	4.080 % 4.877 %	3.324 % 4.013 %	2.569 % 3.232 %	3.825 % 5.096 %
300×300×3	6.165 % 6.427 %	5.912 % 6.416 %	4.649 % 6.238 %	4.358 % 4.849 %	3.210 % 4.664 %	2.490 % 4.171 %	4.672 % 5.184 %
350×350×3	5.264 % 6.568 %	4.763 % 6.381 %	4.314 % 5.773 %	3.891 % 4.402 %	2.669 % 4.322 %	2.411 % 3.158 %	4.588 % 4.918 %
400×400×3	6.091 % 6.702 %	4.707 % 5.413 %	4.599 % 5.201 %	3.706 % 4.168 %	3.444 % 3.993 %	3.214 % 3.654 %	3.849 % 5.168 %
450×450×3	5.880 % 6.429 %	5.710 % 6.376 %	4.441 % 6.514 %	4.195 % 5.349 %	2.711 % 3.934 %	2.088 % 2.605 %	3.585 % 4.323 %
500×500×3	5.234 % 6.260 %	5.941 % 6.412 %	4.234 % 4.859 %	3.789 % 4.206 %	3.234 % 3.842 %	2.272 % 2.614 %	4.196 % 4.781 %

Таблица 4.6 – Результаты сравнения разных наборов управляющих параметров для разработанных метаэвристических алгоритмов (по критерию оценки относительной погрешности решений). Серия экспериментов №2

Размерность задачи	**Алгоритм**						
	ЭА	**ИО**	**ПЗ**	**МС**	**АСО**	**ВСО**	**АВС**
50×50×3	10.439 % 11.719 %	8.851 % 12.641 %	8.444 % 11.489 %	7.787 % 12.958 %	7.650 % 11.504 %	6.666 % 8.537 %	8.296 % 10.315 %
100×100×3	9.668 % 12.768 %	9.368 % 13.738 %	9.260 % 10.912 %	8.962 % 11.838 %	7.069 % 11.794 %	6.264 % 11.480 %	8.689 % 11.847 %
150×150×3	9.958 % 12.750 %	10.510 % 11.638 %	10.074 % 10.596 %	7.928 % 10.423 %	7.258 % 11.462 %	7.634 % 9.156 %	8.470 % 11.435 %
200×200×3	10.401 % 14.889 %	9.901 % 11.945 %	8.813 % 12.058 %	9.435 % 10.583 %	7.189 % 9.925 %	7.247 % 11.364 %	8.898 % 11.283 %
250×250×3	10.580 % 12.761 %	9.069 % 13.347 %	9.331 % 10.530 %	7.868 % 10.805 %	6.815 % 9.882 %	5.914 % 11.214 %	8.120 % 12.750 %
300×300×3	9.223 % 11.851 %	9.106 % 11.681 %	9.569 % 11.331 %	8.832 % 11.724 %	7.917 % 11.841 %	7.808 % 8.071 %	8.919 % 10.629 %
350×350×3	10.145 % 13.561 %	10.089 % 14.287 %	9.468 % 13.217 %	9.005 % 9.527 %	8.013 % 9.091 %	7.523 % 10.397 %	9.512 % 12.253 %
400×400×3	10.048 % 13.216 %	10.113 % 11.414 %	8.486 % 10.913 %	7.946 % 11.969 %	7.741 % 9.213 %	6.546 % 10.946 %	9.495 % 11.099 %
450×450×3	10.177 % 13.789 %	10.817 % 14.477 %	9.258 % 13.445 %	8.652 % 12.808 %	8.031 % 12.171 %	6.149 % 9.761 %	8.940 % 11.334 %
500×500×3	10.306 % 11.583 %	9.680 % 13.543 %	10.054 % 12.045 %	7.763 % 9.878 %	7.400 % 9.844 %	7.298 % 9.852 %	8.788 % 12.393 %

Таблица 4.7 – Результаты сравнения разных наборов управляющих параметров для разработанных метаэвристических алгоритмов (по критерию оценки среднего времени сходимости). ε = 5 %

Размерность задачи	**Алгоритм**						
	ЭА	**ИО**	**ПЗ**	**МС**	**АСО**	**ВСО**	**АВС**
50×50×3	373.56 с 437.158 с	331.906 с 366.582 с	304.417 с 343.220 с	181.681 с 276.395 с	152.454 с 276.316 с	158.190 с 228.507 с	271.270 с 333.072 с
100×100×3	671.094 с 787.170 с	702.297 с 776.442 с	648.246 с 764.552 с	540.074 с 599.052 с	420.476 с 538.063 с	268.096 с 395.389 с	424.160 с 668.871 с
150×150×3	1098.810 с 1169.205 с	998.428 с 1095.509 с	794.571 с 911.623 с	560.398 с 708.727 с	632.205 с 658.600 с	553.170 с 759.127 с	655.548 с 897.921 с
200×200×3	1460.274 с 1596.321 с	1171.629 с 1400.324 с	1108.204 с 1368.960 с	842.907 с 911.899 с	689.870 с 923.739 с	523.700 с 876.545 с	840.677 с 1261.438 с
250×250×3	1821.943 с 2109.135 с	1563.514 с 1844.811 с	1574.253 с 1751.089 с	1224.586 с 1463.541 с	997.311 с 1204.723 с	771.305 с 970.265 с	1148.216 с 1529.318 с
300×300×3	2219.967 с 2314.173 с	2129.306 с 2309.870 с	1674.521 с 2245.951 с	1568.952 с 1746.206 с	1156.139 с 1679.275 с	897.176 с 1502.089 с	1682.107 с 1867.228 с
350×350×3	2211.775 с 2759.121 с	2001.036 с 2680.897 с	1812.538 с 2424.821 с	1634.364 с 1849.309 с	1121.890 с 1815.563 с	1013.063 с 1326.932 с	1927.455 с 2066.513 с
400×400×3	2924.414 с 3217.462 с	2259.698 с 2598.877 с	2208.377 с 2496.695 с	1779.604 с 2000.876 с	1653.795 с 1917.122 с	1543.654 с 1753.958 с	1847.761 с 2481.415 с
450×450×3	3175.401 с 3472.270 с	3084.304 с 3443.425 с	2399.066 с 3518.112 с	2265.746 с 2889.060 с	1464.783 с 2124.620 с	1127.695 с 1407.299 с	1935.915 с 2335.291 с
500×500×3	3140.872 с 3756.025 с	3565.440 с 3847.244 с	2540.999 с 2916.393 с	2273.515 с 2524.119 с	1941.085 с 2306.082 с	1364.098 с 1568.443 с	2518.498 с 2869.471 с

Таблица 4.8 – Результаты сравнения разных наборов управляющих параметров для разработанных метаэвристических алгоритмов (по критерию оценки среднего времени сходимости). ε = 10 %

Размерность задачи	**Алгоритм**						
	ЭА	**ИО**	**ПЗ**	**МС**	**АСО**	**ВСО**	**АВС**
50×50×3	5.220 с 5.860 с	4.426 с 6.321 с	4.222 с 5.745 с	3.894 с 6.479 с	3.825 с 5.752 с	3.333 с 4.269 с	4.148 с 5.158 с
100×100×3	9.668 с 12.768 с	9.368 с 13.738 с	9.260 с 10.912 с	8.962 с 11.838 с	7.069 с 11.794 с	6.264 с 11.480 с	8.689 с 11.847 с
150×150×3	14.937 с 19.125 с	15.765 с 17.457 с	15.111 с 15.894 с	11.892 с 15.635 с	10.887 с 17.193 с	11.451 с 13.734 с	12.705 с 17.153 с
200×200×3	20.802 с 29.778 с	19.802 с 23.890 с	17.626 с 24.116 с	18.870 с 21.166 с	14.378 с 19.850 с	14.494 с 22.728 с	17.796 с 22.566 с
250×250×3	26.450 с 31.903 с	22.673 с 33.368 с	23.328 с 26.325 с	19.670 с 27.013 с	17.038 с 24.705 с	14.785 с 28.035 с	20.300 с 31.875 с
300×300×3	27.669 с 35.553 с	27.318 с 35.043 с	28.707 с 33.993 с	26.496 с 35.172 с	23.751 с 35.523 с	23.424 с 24.213 с	26.757 с 31.887 с
350×350×3	35.508 с 47.464 с	35.312 с 50.005 с	33.138 с 46.260 с	31.518 с 33.345 с	28.046 с 31.819 с	26.331 с 36.390 с	33.292 с 42.886 с
400×400×3	40.192 с 52.864 с	40.452 с 45.656 с	33.944 с 43.652 с	31.784 с 47.876 с	30.964 с 36.852 с	26.184 с 43.784 с	37.980 с 44.396 с
450×450×3	45.797 с 62.051 с	48.677 с 65.147 с	41.661 с 60.503 с	38.934 с 57.636 с	36.140 с 54.770 с	27.671 с 43.925 с	40.230 с 51.003 с
500×500×3	51.530 с 57.915 с	48.400 с 67.715 с	50.270 с 60.225 с	38.815 с 49.390 с	37.000 с 49.220 с	36.490 с 49.260 с	43.940 с 61.965 с

Таблица 4.9 – Результаты сравнения разных наборов управляющих параметров для разработанных метаэвристических алгоритмов (по критерию оценки доли успешных запусков)

Размерность задачи	**Алгоритм**						
	ЭА	**ИО**	**ПЗ**	**МС**	**АСО**	**ВСО**	**АВС**
50×50×3	97.184 % 96.129 %	97.875 % 97.291 %	98.330 % 97.694 %	99.621 % 98.798 %	99.131 % 98.807 %	99.236 % 99.607 %	98.887 % 97.861 %
100×100×3	97.815 % 96.842 %	97.550 % 96.933 %	97.998 % 97.036 %	98.902 % 98.413 %	99.904 % 98.917 %	98.833 % 99.892 %	99.869 % 97.831 %
150×150×3	97.296 % 96.905 %	97.858 % 97.316 %	98.987 % 98.339 %	99.709 % 99.466 %	99.893 % 99.743 %	99.673 % 99.188 %	99.761 % 98.416 %
200×200×3	97.318 % 96.752 %	98.520 % 97.569 %	98.784 % 97.697 %	99.892 % 99.602 %	99.472 % 99.554 %	98.782 % 99.749 %	99.898 % 98.145 %
250×250×3	97.328 % 96.370 %	98.189 % 97.251 %	98.154 % 97.565 %	99.320 % 98.523 %	99.924 % 99.387 %	99.169 % 99.832 %	99.575 % 98.304 %
300×300×3	97.235 % 96.973 %	97.488 % 96.984 %	98.751 % 97.162 %	99.042 % 98.551 %	99.810 % 98.736 %	99.090 % 99.229 %	98.728 % 98.216 %
350×350×3	98.136 % 96.832 %	98.637 % 97.019 %	99.086 % 97.627 %	99.509 % 98.998 %	99.269 % 99.078 %	99.011 % 99.758 %	98.812 % 98.482 %
400×400×3	97.309 % 96.698 %	98.693 % 97.987 %	98.801 % 98.199 %	99.694 % 99.232 %	99.956 % 99.407 %	99.814 % 99.746 %	99.551 % 98.232 %
450×450×3	97.520 % 96.971 %	97.690 % 97.024 %	98.959 % 96.886 %	99.205 % 98.051 %	99.311 % 99.466 %	98.688 % 99.205 %	99.815 % 99.077 %
500×500×3	98.166 % 97.140 %	97.459 % 96.988 %	99.166 % 98.541 %	99.611 % 99.194 %	99.834 % 99.558 %	98.872 % 99.214 %	99.204 % 98.619 %

4.3 Выводы

1. Разработана структура программного комплекса поддержки принятия решений, реализующего алгоритмы и методы оптимизации размещения элементов развивающихся информационных систем и обеспечивающего уменьшение затрат на создание и модернизацию беспроводной сети передачи данных и увеличение эффективности мероприятий по планированию и построению беспроводных информационных систем, отличительной особенностью которой является наличие инструментов автоматизации проведения серий вычислительных экспериментов с целью оценки качества алгоритмов, лежащих в основе созданной программной системы.

2. На базе созданного программного комплекса поддержки принятия решений был проведен ряд вычислительных экспериментов по сравнению предложенных в главе 2 алгоритмов, а также эксперименты, направленные на исследование эффективности метода, предложенного в главе 3.

3. Предложенные в главе 2 метаэвристические алгоритмы позволяют эффективно решать задачу размещения элементов при создании и модернизации развивающихся информационных систем (на примере беспроводных сетей передачи данных).

4. На задачах малой размерности предложенные в главе 2 алгоритмы обеспечивают получение точных значений целевой функции практически при каждом запуске алгоритма. При этом предложенные алгоритмы справляются с решением задачи размещения элементов развивающихся информационных систем за очень малое время, в отличие от точного метода полного перебора.

5. Предложенные в главе 2 алгоритмы поиска с запретами, имитации отжига и мультистарта справляются с решением задачи размещения элементов развивающихся информационных систем лучше, чем алгоритм локального спуска.

6. Алгоритмы роевого интеллекта – пчелиный алгоритм BCO и муравьиный алгоритм – в среднем лучше других алгоритмов справляются с решением задачи размещения элементов развивающихся информационных систем.

7. Решение задачи планирования и оптимизации информационных систем (на примере беспроводных сетей передачи данных) с использованием алгоритмов с управляющими параметрами, найденными при помощи предлагаемого в главе 3 метода, позволяет значительно улучшить результаты по сравнению с использованием алгоритмов со значениями параметров «по умолчанию». Данное улучшение способно в среднем на несколько процентов уменьшить затраты на создание сети, а значит, увеличивает эффективность мероприятий по планированию и построению беспроводных информационных систем.

ЗАКЛЮЧЕНИЕ

1. Предложена математическая модель задачи размещения элементов развивающихся информационных систем, отличающаяся учетом возможности модернизации элементов системы, свойств мест-кандидатов, помех при межмодульном взаимодействии элементов системы и подразумевающая использование нескольких типов элементов информационной системы. Данная задача формализована как NP-трудная задача размещения устройств обслуживания с ограничениями на мощности, что позволило обосновать целесообразность использования эвристических алгоритмов.

2. Разработаны метаэвристические алгоритмы (эволюционный, имитации отжига, поиск с запретами, мультистарта, муравьиный, пчелиный) интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем. Отличительной особенностью предлагаемых алгоритмов является: для алгоритмов локального поиска – построение окрестности решения при помощи небольших операций, вносящих изменения в конфигурацию информационной системы; для муравьиного алгоритма – использование двух колоний искусственных муравьев для двухфазного решения задачи.

3. Предложен метод параметрической метаоптимизации разработанных метаэвристических алгоритмов на базе эволюционного подхода, отличающийся возможностью одновременной настройки вещественных, целочисленных и символьных управляющих параметров и обеспечивающий значительное улучшение производительности алгоритмов решения задачи размещения элементов развивающихся информационных систем.

4. Разработана структура программного комплекса интеллектуальной поддержки принятия решений, реализующего методы и алгоритмы оптимизации размещения элементов развивающихся информационных систем и обеспечивающего уменьшение затрат и увеличение эффективности мероприятий

по планированию и модернизации информационных систем, отличительной особенностью которой является наличие инструментов автоматизации проведения серий вычислительных экспериментов с целью оценки качества алгоритмов, лежащих в основе программного комплекса.

5. Элементы программного обеспечения прошли государственную регистрацию в Роспатенте.

6. Отдельные модули разработанного программного комплекса были внедрены в ОАО "Мобильные ТелеСистемы" (филиал в Липецкой области) и АО "ЭР-Телеком Холдинг" (филиал в г. Липецк).

СПИСОК ЛИТЕРАТУРЫ

1. Вишневский В.М., Ляхов А.И., Портной С.Л., Шахнович И.В. Широкополосные беспроводные сети передачи информации. – М.: Техносфера, 2005. – 592 с.

2. Вишневский В.М. Теоретические основы проектирования компьютерных сетей. – М.: Техносфера, 2003. – 512 с.

3. Волкова В.Н., Денисов А.А. Теория систем и системный анализ. – 2-е изд., перераб и доп. – М.: Юрайт, 2014. – 616 с.

4. Глушко С.И. Иерархические нечеткие многоколониальные муравьиные алгоритмы и комплекс программ оптимизации телекоммуникационных сетей нефтетранспортных предприятий: дис. ... кандидата технических наук. – РХТУ, Москва, 2013. – 145 с.

5. Гончаров Е.Н., Кочетов Ю.А. Вероятностный поиск с запретами для дискретных задач безусловной оптимизации. // Дискретный анализ и исследование операций. 2002. №2. – С. 13-30.

6. Гончаров Е.Н., Кочетов Ю.А. Поведение вероятностных жадных алгоритмов для многостадийной задачи размещения. // Дискретный анализ и исследование операций. 1999. №1. – С. 12-32.

7. ГОСТ Р 55897-2013 Сети подвижной радиосвязи. Зоны обслуживания. Методы расчета. – М.: Стандартинформ, 2014 – 20 с.

8. Гришин А.А., Карпенко А.П. Исследование эффективности метода пчелиного роя в задаче глобальной оптимизации. // Наука и образование. 2010. №8. – С. 1-28.

9. Гэри М. Джонсон Д. Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982. – 419 с.

10. Дворецкий С.И., Матвеев С.В., Путин С.Б., Туголуков Е.Н. Основы математического моделирования и оптимизации процессов и систем очистки и

регенерации воздуха: учебное пособие. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2008. – 324 с.

11. Демидова Л.А., Клюева И.А. Разработка и исследование гибридных версий алгоритма роя частиц на основе алгоритмов поиска по сетке. // Вестник Рязанского государственного радиотехнического университета. 2016. №3 (57). – С. 105-116.

12. Демидова Л.А., Петрова Н.А. Применение эволюционного подхода к задачам оптимизации параметров сложных технических систем Computers & Industrial Engineering. // Вестник Рязанского государственного радиотехнического университета. 2013. №3 (45). – С. 93-100.

13. Дорохов И.Н., В.В. Меньшиков. Системный анализ процессов химической технологии. – М.: Наука, 2005. – 584 с.

14. Елизаров Д.Э., Бурковский В.Л. Алгоритм решения задачи оптимального размещения узлов обслуживания в условиях развивающихся мультисервисных сетей. // Вестник Воронежского государственного технического университета. 2016. Т. 12. №3. – С. 4-7.

15. Елизаров Д.Э., Бурковский В.Л., Воропаев А.П. Обобщенная оптимизационная модель развития мультисервисных сетей. // Вестник Воронежского государственного технического университета. 2015. Т. 11. №3. – С. 28-30.

16. Елизаров Д.Э., Бурковский В.Л. Оптимизационные модели формирования структуры развивающихся мультисервисных сетей информационного обслуживания населения. // Вестник Воронежского государственного технического университета. 2015. Т. 11. №4. – С. 20-23.

17. Ермолаев С.Ю. Оптимальное размещение базовых станций. // Telecommunication Sciences. 2010. №1. – С. 30-36.

18. Ермолаев С.Ю. Разработка алгоритмов размещения базовых станций на основе методов оптимизации для сетей беспроводного доступа: дис. ... кандидата технических наук. – ПГУТИ, Самара, 2010. – 163 с.

19. Задача размещения с ограничениями на мощности [Электронный ресурс]. Режим доступа: http://www.math.nsc.ru/AP/benchmarks/CFLP/cflp.html, свободный (дата обращения: 01.06.2017).

20. Интрон – Википедия [Электронный ресурс]. Режим доступа: http://ru.wikipedia.org/wiki/Интрон, свободный (дата обращения: 01.06.2017).

21. Информационная система [Электронный ресурс]. Режим доступа: http://dic.academic.ru/dic.nsf/ruwiki/101791, свободный (дата обращения: 01.06.2017).

22. Карпенко А.П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учебное пособие. – М.: Издательство МГТУ им. Н.Э. Баумана, 2014. – 446 с.

23. Когаловский М.Р. Перспективные технологии информационных систем. – М.: ДМК Пресс, 2003. – 288 с.

24. 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: МГУ. – 2001. – С. 87-117.

25. Кочетов Ю.А. Двухуровневые задачи размещения. // Труды ИВМ и МГ СО РАН: Серия Информатика. 2007. №7. – С. 97-104.

26. Кочетов Ю.А. Методы локального поиска для дискретных задач размещения: дис. ... доктора физико-математических наук. – ИМ СО РАН, Новосибирск, 2009. – 267 с.

27. Кравец О.Я., Ачкасов А.В. Межмодульное взаимодействие в распределенных информационных системах с нестабильной структурой. // Информационные технологии моделирования и управления. 2014. №2 (86). – С. 146-157.

28. Лореш М.А. Разработка и исследование алгоритмов муравьиной колонии для решения задач оптимального размещения предприятий. – ОмГУ, Омск, 2006. – 113 с.

29. Малыш В.Н., Скаков Е.С. Эволюционный алгоритм нахождения оптимальных значений управляющих параметров для метаэвристических методов решения задачи планирования беспроводной сети. // Вестник Рязанского государственного радиотехнического университета. 2016. №2 (56). – С. 64-72.

30. Матвеенко И.М., Бурковский В.Л. Модели анализа телекоммуникационных сетей в процессе их развития. // Вестник Воронежского государственного технического университета. 2008. Т. 4. №5. – С. 85-87.

31. Матвеенко И.М., Бурковский В.Л. Модели оптимизации параметров обслуживания в развивающихся корпоративных сетях специального назначения. // Вестник Воронежского государственного технического университета. 2007. Т. 3. №5. – С. 73-76.

32. Матвеенко И.М., Бурковский В.Л. Модели развивающихся распределенных информационных систем на основе моделей социальных систем. // Вестник Воронежского государственного технического университета. 2008. Т. 4. №5. – С. 75-77.

33. Матвеенко И.М. Структурное моделирование развивающихся информационных систем обслуживания клиентов телекоммуникационных сетей: дис. ... кандидата технических наук. – ВГТУ, Воронеж, 2008. – 153 с.

34. Метод ветвей и границ [Электронный ресурс]. Режим доступа: http://www.math.nsc.ru/AP/benchmarks/UFLP/uflp_bb.html, свободный (дата обращения: 01.06.2017).

35. Метод Лагранжевых релаксаций [Электронный ресурс]. Режим доступа: http://www.math.nsc.ru/AP/benchmarks/UFLP/ufpl_lag.html, свободный (дата обращения: 01.06.2017).

36. Оператор ветвления – Википедия [Электронный ресурс]. Режим доступа: http://ru.wikipedia.org/wiki/Оператор_ветвления, свободный (дата обращения: 01.06.2017).

37. Особенности построения сетей широкополосного доступа формата WiMAX [Электронный ресурс]. Режим доступа:

http://cyberleninka.ru/article/n/osobennosti-postroeniya-setey-shirokopolosnogo-dostupa-formata-wimax, свободный (дата обращения: 01.06.2017).

38. Пантелеев А.В. Метаэвристические алгоритмы поиска глобального экстремума. – М.: Изд-во МАИ-ПРИНТ, 2009. – 160 с.

39. Простейшая задача размещения [Электронный ресурс]. Режим доступа: http://www.math.nsc.ru/AP/benchmarks/UFLP/uflp.html, свободный (дата обращения: 01.06.2017).

40. Руднев А.С. Вероятностный поиск с запретами для задачи упаковки кругов и прямоугольников в полосу. // Дискретный анализ и исследование операций. 2009. №4. – С. 61-86.

41. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. – М.: Горячая линия-Телеком, 2006. – 452 с.

42. Саати Т., Кернс К. Аналитическое планирование. Организация систем. / Пер. с англ. Р.Г. Вачнадзе; под редакцией И.А. Ушакова. – М.: Радио и связь, 1991. – 224 с.

43. Скаков Е.С. Алгоритм имитации отжига для решения оптимизационных задач. // Сборник материалов V Международной научно-практической конференции «Актуальные проблемы технических наук» – Уфа, 2015. – С. 56-60.

44. Скаков Е.С., Малыш В.Н. Алгоритм имитации отжига в задаче оптимизации размещения базовых станций. // Системы управления и информационные технологии. 2015. №2 (60). – С. 90-94.

45. Скаков Е.С., Малыш В.Н. Алгоритмическое обеспечение размещения базовых станций при планировании беспроводных сетей передачи данных. // 18-я Международная научно-техническая конференция «Проблемы передачи и обработки информации в сетях и системах телекоммуникаций». Тезисы докладов – Рязань: РГРТУ, 2015. – С. 190-194.

46. Скаков Е.С., Малыш В.Н. Использование алгоритмов мультистарта и поиска с запретами для решения задачи размещения базовых станций. // Информационно-управляющие системы. 2015. №3 (76). – С. 99-106.

47. Скаков Е.С., Малыш В.Н. Метаэвристики, основанные на муравьином алгоритме оптимизации, для решения задачи планирования беспроводной сети. // III Международная конференция «Устойчивость и процессы управления». Тезисы докладов – Санкт-Петербург, 2015. – С. 331-332.

48. Скаков Е.С., Малыш В.Н. Модифицированный алгоритм пчелиной колонии ABC для проектирования топологии беспроводной сети. // Труды Международного симпозиума «НАДЕЖНОСТЬ И КАЧЕСТВО» – Пенза, 2016. Т. 1. С. 293-296.

49. Скаков Е.С., Малыш В.Н. Муравьиный алгоритм для решения задачи размещения с ограничениями на мощности. // Сборник материалов Международной летней научной школы «ПАРАДИГМА» – Болгария, Варна: Изд-во "ЦЕНТЪР ЗА НАУЧНИ ИЗСЛЕДВАНИЯ И ИНФОРМАЦИЯ "ПАРАДИГМА"" ЕООД БЪЛГАРИЯ, 2015. – С. 167-174.

50. Скаков Е.С., Малыш В.Н. Постановка модифицированной задачи размещения базовых станций. // Сборник материалов Международной научно-практической конференции «ИНФОРМАЦИОННЫЕ УПРАВЛЯЮЩИЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (ИУСТ-ОДЕССА-2015) – Одесса, 2015. – С. 114-117.

51. Скаков Е.С., Малыш В.Н. Применение генетического алгоритма для решения задачи проектирования беспроводной сети. // Вестник Липецкого государственного педагогического университета. Серия МИФЕ. 2015. №1 (16). – С. 59-63.

52. Скаков Е.С., Малыш В.Н. Применение эволюционного подхода для нахождения оптимальных значений управляющих параметров метаэвристических алгоритмов. // Сборник материалов Международной научно-практической

конференции «ИНФОРМАЦИОННЫЕ УПРАВЛЯЮЩИЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (ИУСТ-ОДЕССА-2016) – Одесса, 2016. – С. 297-299.

53. Скаков Е.С., Малыш В.Н. Пчелиный алгоритм оптимизации для решения задачи планирования беспроводной сети. // Программные продукты и системы. 2016. №3 (115). – С. 67-73.

54. Скаков Е.С., Малыш В.Н. Решение задачи размещения центров обработки данных при помощи оптимизации подражанием муравьиной колонии. // Системы управления и информационные технологии. 2017. №1 (67). – С. 42-47.

55. Скаков Е.С., Малыш В.Н. Эволюционный алгоритм решения задачи размещения базовых станций. // Вестник Рязанского государственного радиотехнического университета. 2015. №1 (51). – С. 46-52.

56. Скаков Е.С. Метод поиска с запретами для решения оптимизационных задач. // Сборник материалов XV Международной научно-практической конференции «Новое слово в науке и практике: гипотезы и апробация результатов исследований» – Новосибирск, 2015. – С. 166-171.

57. Скаков Е.С. Проблема оптимизации размещения базовых станций в сетях стандарта LTE. // Сборник материалов Международной научно-практической конференции «ИНФОРМАЦИОННЫЕ УПРАВЛЯЮЩИЕ СИСТЕМЫ И ТЕХНОЛОГИИ» (ИУСТ-ОДЕССА-2014) – Одесса, 2014. – С. 248-251.

58. Скаков Е.С. Программа для решения задачи планирования беспроводных сетей передачи данных при помощи метаэвристических методов оптимизации. М.: Роспатент. – Свидетельство №2016613530 от 29.03.2016.

59. Скаков Е.С. Программа для решения задачи размещения базовых станций методами поиска с запретами и мульти-старта. М.: Роспатент. – Свидетельство №2015618283 от 05.08.2015.

60. Скаков Е.С. Решение оптимизационных задач при помощи метода мульти-старта. // Сборник докладов V Международной научно-практической

конференции «Актуальные проблемы технических наук». Том 1 – Тамбов: ООО «Консалтинговая компания Юком», 2015. – С. 139-142.

61. Скаков Е.С. Эволюционный алгоритм для решения задачи размещения устройств обслуживания. // Математическое и программное обеспечение вычислительных систем: Межвуз. сб. науч. тр. / под ред. А.Н. Пылькина - Рязань: РГРТУ, 2014. С.103-110.

62. Скаков Е.С. Эволюционный алгоритм решения задачи размещения базовых станций. М.: Роспатент. – Свидетельство №2015611925 от 09.02.2015.

63. Сурмин Ю.П. Теория систем и системный анализ: учебное пособие. – Киев: МАУП, 2003. – 368 с.

64. Теория систем и системный анализ в управлении организациями: Справочник: учебное пособие / Под редакцией В.Н. Волковой и А.А. Емельянова. – М.: Финансы и статистика, 2006. – 848 с.

65. Усманова А.Р. Метод поиска с запретами для задач упаковки в контейнеры: дис. ... кандидата физико-математических наук. – УГАТУ, Уфа, 2002. – 101 с.

66. Федеральный закон от 27.07.2006 N 149-ФЗ (ред. от 19.12.2016) "Об информации, информационных технологиях и о защите информации" (с изм. и доп., вступ. в силу с 01.01.2017) [Электронный ресурс]. Режим доступа: http://www.consultant.ru/cons/cgi/online.cgi?req=doc&base=LAW&n=200126&fld=134&dst=1000000001,0&rnd=0.6628550026339017#0, свободный (дата обращения: 01.06.2017).

67. Хабаров Е.Н., Кравец О.Я. Особенности оптимального выбора алгоритмов межмодульного взаимодействия компонент распределенных программных систем. // Вестник Воронежского государственного технического университета. 2011. №6. – С. 180-184.

68. Цикл (программирование) – Википедия [Электронный ресурс]. Режим доступа: http://ru.wikipedia.org/wiki/Цикл_(программирование), свободный (дата обращения: 01.06.2017).

69. Чернышов В.Н., Чернышов А.В. Теория систем и системный анализ: учебное пособие. – Тамбов: Изд-во Тамб. гос. техн. ун-та, 2008. – 96 с.

70. Широков В. Методология создания беспроводных мультисервисных сетей класса WiMAX. Часть 1. // Технологии и средства связи. 2010. №1. – С. 32-33.

71. Штовба С.Д. Муравьиные алгоритмы. // Exponenta Pro. Математика в приложениях. 2003. №4. – С. 70-75.

72. Щербина О.А. Метаэвристические алгоритмы для задач комбинаторной оптимизации (обзор). // Таврический Вестник Информатики и Математики. 2014. №1. – С. 56-72.

73. Ahmad F., Isa N.A.M., Hussain Z., Osman M.K., Sulaiman S.N. A GA-Based Feature Selection and Parameter Optimization of an ANN in Diagnosing Breast Cancer. // Pattern Analysis and Applications. 2015. Vol. 18 (4) – pp. 861-870.

74. Amaldi E., Belotti P., Capone A., Malucelli F. Optimizing Base Station Location and Configuration in UMTS Networks. // Annals of Operations Research. 2006. Vol. 146 (1) – pp. 135-151.

75. Amaldi E., Capone A., Malucelli F. Planning UMTS Base Station Location: Optimization Models with Power Control and Algorithms. // IEEE Transactions on Wireless Communications. 2003. Vol. 2 (5) – pp. 939-952.

76. Amaldi E., Capone A., Malucelli F. Radio Planning and Coverage Optimization of 3G Cellular Networks. // Wireless Networks. 2008. Vol. 14 (4) – pp. 435-447.

77. Amaldi E., Capone A., Malucelli F., Signori F. Optimization Models and Algorithms for Downlink UMTS Radio Planning. // Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2003). New Orleans, USA. 2003, March. – pp. 133-137.

78. Bao L., Zeng J. Comparison and Analysis of the Selection Mechanism in the Artificial Bee Colony Algorithm. // Proceedings of the 2009 9th International

Conference on Hybrid Intelligent Systems (HIS 2009). Shenyang, China. 2009, August. – pp. 411-416.

79. Boese K.D., Kahng A.B., Muddu S. A New Adaptive Multi-Start Technique for Combinatorial Global Optimizations. // Operations Research Letters. 1994. Vol. 16 (2) – pp. 101-113.

80. Bonabeau E., Dorigo M., Theraulaz G. Swarm Intelligence: from Natural to Artificial Systems. – Oxford University Press, 1999. – 320 p.

81. Booker L.B., Goldberg D.E., Holland J.H. Classifier Systems and Genetic Algorithms. // Artificial Intelligence. 1989. Vol. 40 (1) – pp. 235-282.

82. Bräysy O., Hasle G., Dullaert W. A Multi-Start Local Search Algorithm for the Vehicle Routing Problem with Time Windows. // European Journal of Operational Research. 2004. Vol. 159 (3) – pp. 586-605.

83. Bremermann H.J. The Evolution of Intelligence: The Nervous System as a Model of its Environment. Technical report no.1. – University of Washington, USA, 1958. – 99 p.

84. Brest J., Greiner S., Boskovic B., Mernik M., Zumer V. Self-Adapting Control Parameters in Differential Evolution: a Comparative Study on Numerical Benchmark Problems. // IEEE Transactions on Evolutionary Computation. 2006. Vol. 10 (6) – pp. 646-657.

85. Cachón A., Vázquez R.A. Tuning the Parameters of an Integrate and Fire Neuron via a Genetic Algorithm for Solving Pattern Recognition Problems. // E Neurocomputing. 2015. Vol. 148 – pp. 187-197.

86. Černý V. Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm. // Journal of Optimization Theory and Applications. 1985. Vol. 45 (1) – pp. 41-51.

87. Chang W.D. Nonlinear System Identification and Control Using a Real-Coded Genetic Algorithm. // Applied Mathematical Modelling. 2007. Vol. 31 (3) – pp. 541-550.

88. Cochennec J.Y. Activities On Next-Generation Networks Under Global Information Infrastructure in ITU-T. // IEEE Communications Magazine. 2002. Vol. 40 (7) – pp. 98-101.

89. COST231 final report, COST Action 231, Digital Mobile Radio Towards Future Generation Systems, European Commission/COST Telecommunications, Brussels, Belgium, 1999. – 516 p.

90. Das H., Cummings P.T., Le Van M.D. Scheduling of Serial Multiproduct Batch Processes via Simulated Annealing. // Computers & Chemical Engineering. 1990. Vol. 14 (12) – pp. 1351-1362.

91. Daskin M.S. Network and Discrete Location: Models, Algorithms, and Applications. – Wiley, 2013. – 536 p.

92. Daskin M.S., Owen S.H. Strategic Facility Location: A Review. // European Journal of Operational Research. 1998. Vol. 111 (3) – pp. 423-447.

93. Daskin M.S, Revelle C.S., Eiselt H.A. A Bibliography for Some Fundamental Problem Categories in Discrete Location Science. // European Journal of Operational Research. 2008. Vol. 184 (3) – pp. 817-848.

94. Davidović T., Ramljak D., Šelmić M., Teodorović D. Bee Colony Optimization for the p-center Problem. // Computers & Operations Research. 2011. Vol. 38 (10) – pp. 1367-1376.

95. Davis W.S., Yen D.C. The Information System Consultant's Handbook: Systems Analysis and Design. – CRC Press, 1998. – 800 p.

96. Dianati M., Song I., Treiber M. An Introduction to Genetic Algorithms and Evolution Strategies. Technical Report. – University of Waterloo, Ontario, Canada, 2002 – 11 p.

97. Dorigo M., Birattari M., Stützle T. Ant Colony Optimization. // IEEE Computational Intelligence Magazine. 2006. Vol. 1 (4) – pp. 28-39.

98. Dorigo M., Bonabeau E., Theraulaz G. Ant algorithms and Stigmergy. // Future Generation Computer Systems. 2000. Vol. 16 (8) – pp. 851-871.

99. Dorigo M., Colorni A., Maniezzo V. Distributed optimization by Ant Colonies. // Proceedings of the 1st European Conference on Artificial Life. Paris, France. 1991, December. – pp. 134-142.

100. Dorigo M., Gambardella L.M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. // IEEE Transactions on Evolutionary Computation. 1997. Vol. 1 (1) – pp. 53-66.

101. Dorigo M. Optimization, Learning and Natural Algorithms [in Italian]: Ph. D. Thesis. – Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

102. Dréo J., Petrowski A., Siarry P., Taillard E. Metaheuristics for Hard Optimization: Methods and Case Studies. – Springer, 2006. – 382 p.

103. Drezner Z., Hamacher H.W. Facility Location: Applications and Theory. – Springer, 2002. – 460 p.

104. Eberspächer J., Vögel H.J., Bettstetter C., Hartmann C. GSM – Architecture, Protocols and Services. – Wiley, 2008. – 338 p.

105. Eiben A.E., Smit S.K. Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms. // Swarm and Evolutionary Computation. 2011. Vol. 1 (1) – pp. 19-31.

106. Faigle U., Kern W. Some Convergence Results for Probabilistic Tabu Search. // ORSA Journal on Computing. 1992. Vol. 4 (1) – pp. 32-37.

107. Fodor G., Koutsimanis C., Rácz A., Reider N., Simonsson A., Müller W. Intercell Interference Coordination in OFDMA Networks and in the 3GPP Long Term Evolution System. // Journal of Communications. 2009. Vol. 4 (7) – pp. 445-453.

108. Gaertner D., Clark K. On Optimal Parameters for Ant Colony Optimization Algorithms. // Proceedings of the 2005 International Conference on Artificial Intelligence. Las Vegas, USA. 2005, June. – pp. 83-89.

109. Gambardella L.M., Taillard É., Agazzi G. MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. // New Ideas in Optimization. – McGraw-Hill Ltd., 1999. – pp. 63-76.

110. Gendreau M., Potvin J.Y. Handbook of Metaheuristics. – Springer, 2010. – 669 p.

111. Glover F. Future Paths for Integer Programming and Links to Artificial Intelligence. // Computers and Operations Research. 1986. Vol. 13 (5) – pp. 533-549.

112. Glover F. Tabu Search—Part I. // ORSA Journal on Computing. 1989. Vol. 1 (3) – pp. 190-206.

113. Glover F. Tabu Search—Part II. // ORSA Journal on Computing. 1990. Vol. 2 (1) – pp. 4-32.

114. González-Brevis P., Gondzio J., Fan Y., Poor H.V., Thompson J., Krikidis I., Chung P.J. Base Station Location Optimization for Minimal Energy Consumption in Wireless Networks. // Proceedings of the 73rd IEEE Vehicular Technology Conference (VTC Spring). Budapest, Hungary. 2011, May. – pp. 1-5.

115. Hata M. Empirical Formula for Propagation Loss in Land Mobile Radio Services. // IEEE Transactions on Vehicular Technology. 1980. Vol. 29 (3) – pp. 317-325.

116. Hierarchical Network Design Overview [Электронный ресурс]. Режим доступа: http://www.ciscopress.com/articles/article.asp?p=2202410&seqNum=4, свободный (дата обращения: 01.06.2017).

117. Holland J.H. Adaptation in Natural and Artificial Systems. – University of Michigan Press, 1975. – 183 p.

118. Holland J.H. Genetic Algorithms and the Optimal Allocation of Trials. // SIAM Journal on Computing. 1973. Vol. 2 (2) – pp. 88-105.

119. Holland J.H., Goldberg D.E. Genetic Algorithms and Machine Learning. // Machine Learning. 1988. Vol. 3 (2) – pp. 95-99.

120. Holland J.H. Outline for a Logical Theory of Adaptive Systems. // Journal of the Association for Computing Machinery. 1962. Vol. 9 (3) – pp. 297-314.

121. Huang C.L., Wang C.J. A GA-Based Feature Selection and Parameters Optimization for Support Vector Machines. // Expert Systems with Applications. 2006. Vol. 31 (2) – pp. 231-240.

122. Karaboga D., Akay B. A Survey: Algorithms Simulating Bee Swarm Intelligence. // Artificial Intelligence Review. 2009. Vol. 31 (1-4) – pp. 61-85.

123. Karaboga D. An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report TR06. – Erciyes University, Kayseri, Turkey, 2005 – 10 p.

124. Karaboga D., Basturk B. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. // Journal of Global Optimization. 2007. Vol. 39 (3) – pp. 459-471.

125. Karaboga D., Basturk B. On the Performance of Artificial Bee Colony (ABC) Algorithm. // Applied Soft Computing. 2008. Vol. 8 (1) – pp. 687-697.

126. Karaboga D., Gorkemli B., Ozturk C., Karaboga N. A Comprehensive Survey: Artificial Bee Volony (ABC) Algorithm and Applications. // Artificial Intelligence Review. 2014. Vol. 42 (1) – pp. 21-57.

127. Kirkpatrick S., Gelatt C.D., Vecchi M.P. Optimization by Simulated Annealing. // Science. 1983. Vol. 220 (4598) – pp. 671-680.

128. Knightson K., Morita N., Towle T. NGN Architecture: Generic Principles, Functional Architecture, and Implementation. // IEEE Communications Magazine. 2005. Vol. 43 (10) – pp. 49-56.

129. Korte B., Vygen J. Combinatorial Optimization. Theory and Algorithms. – Springer, 2006. – 597 p.

130. Koskie S., Gajic Z. A Nash Game Algorithm for SIR-based Power Control in 3G Wireless CDMA Networks. // IEEE/ACM Transactions on Networking (TON). 2005. Vol. 13 (5) – pp. 1017-1026.

131. Liu S., St-Hilaire M. Cooperative Algorithm for the Global Planning Problem of UMTS Networks. // Proceedings of the Global Communications Conference (GLOBECOM 2010). Miami, USA. 2010, December. – pp. 1-5.

132. Li Z., Li S. LTE Network Planning Based on Game Theory. // Proceedings of the IEEE International Conference on Computer Science and Service System (CSSS 2011). Nanjing, China. 2011, June. – pp. 3963-3966.

133. López-Sánchez A.D., Hernández-Díaz A.G., Vigo D., Caballero R., Molina J. A Multi-Start Algorithm for a Balanced Real-World Open Vehicle Routing Problem. // European Journal of Operational Research. 2014. Vol. 238 (1) – pp. 104-113.

134. Luke S. Essentials of Metaheuristics. – Lulu, 2013. – 242 p.

135. Lućić P., Teodorović D. Computing with Bees: Attacking Complex Transportation Engineering Problems. // International Journal on Artificial Intelligence Tools. 2003. Vol. 12 (3) – pp. 375-394.

136. Marti R., Reinelt G. The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization. – Springer, 2011. – 172 p.

137. Martí R., Resende M.G.C., Ribeiro C.C. Multi-Start Methods for Combinatorial Optimization. // European Journal of Operational Research. 2013. Vol. 226 (1) – pp. 1-8.

138. 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 (6) – pp. 1087-1092.

139. Michallet J., Prins C., Amodeo L., Yalaoui F., Vitry, G. Multi-Start Iterated Local Search for the Periodic Vehicle Routing Problem with Time Windows and Time Spread Constraints on Services. // Computers & Operations Research. 2014. Vol. 41 – pp. 196-207.

140. Millonas M.M. Swarms, Phase Transitions, and Collective Intelligence. // Artificial life III: Addison-Wesley, 1994 – pp. 417-445.

141. Mills K.L., Filliben J.J., Haines A.L. Determining Relative Importance and Effective Settings for Genetic Algorithm Control Parameters. // Wireless Networks. 2008. Vol. 14 (4) – pp. 435-447.

142. Mishra A.R. Advanced Cellular Network Planning and Optimisation: 2G/2.5G/3G... Evolution to 4G. – Wiley, 2007. – 544 p.

143. Mobile WIMAX [Электронный ресурс]. Режим доступа: http://celnet.ru/WIMAX.php, свободный (дата обращения: 01.06.2017).

144. Molisch A.F. Wireless Communications. – Wiley, 2011. – 884 p.

145. Nemhauser G.L., Wolsey L.A. Integer and Combinatorial Optimization. – Wiley, 1999. – 784 p.

146. NGN Working definition [Электронный ресурс]. Режим доступа: http://www.itu.int/ITU-T/studygroups/com13/ngn2004/working_definition.html, свободный (дата обращения: 01.06.2017).

147. Nowakova J., Platos J., Snasel V. Automatic Power System Identification Using Genetic Algorithms. // Proceedings of the 6th International Conference on Intelligent Networking and Collaborative Systems (INCoS-2014). Salerno, Italy. 2014, September. – pp. 133-137.

148. Okumura Y. et al. Field Strength and Its Variability in VHF and UHF Land-Mobile Radio Service. // Review of the Electrical Communication Laboratory. 1968. Vol. 16 (9-10).

149. Papadimitriou C.H., Steiglitz K. Combinatorial Optimization: Algorithms and Complexity. – Prentice-Hall, 1982. – 496 p.

150. Rechenberg I. Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library translation NO. 1122, Farnborough, Hants., UK, August 1965.

151. Reeves C.R. Modern Heuristic Techniques for Combinatorial Problems. – Wiley, 1993. – 320 p.

152. Saily M., Sebire G., Riddington E. GSM/EDGE: Evolution and Performance. – Wiley, 2010. – 504 p.

153. Santi P., Maheshwari R., Resta G., Das S., Blough D.M. Wireless Link Scheduling Under a Graded SINR Interference Model. // Proceedings of the 2nd ACM International Workshop on Foundations of Wireless Ad Hoc and Sensor Networking and Computing. New Orleans, USA. 2009, May. – pp. 3-12.

154. Saunders S.R., Aragón-Zavala A. Antennas and Propagation for Wireless Communication Systems. – Wiley, 2007. – 546 p.

155. Skakov E., Malysh V. Ant Colony Optimization Algorithms for Wireless Network Planning Problem Solving. // Proceedings of the 2015 International Conference "Stability and Control Processes" in Memory of V.I. Zubov (SCP). Saint Petersburg, Russia. 2015, October. – pp. 348-352.

156. Skakov E., Malysh V. Simulated Annealing and Evolutionary Algorithm for Base Station Location Problem: a Comparison of Methods. // Journal of Information Technology and Applications. 2015. Vol. 5 (2) – pp. 88-96.

157. Sörensen K., Glover F. Metaheuristics. // Encyclopedia of Operations Research and Management Science. Springer US. 2013. – pp. 960-970.

158. Sörensen K. Metaheuristics – the Metaphor Exposed. // International Transactions in Operational Research. 2015. Vol. 22 (1) – pp. 3-18.

159. Sridharan R. The Capacitated Plant Location Problem. // European Journal of Operational Research. 1995. Vol. 87 (2) – pp. 203-213.

160. St-Hilaire M., Chinneck J.W., Chamberland S., Pierre S. Efficient Solution of the 3G Network Planning Problem. // Computers & Industrial Engineering. 2012. Vol. 63 (4) – pp. 819-830.

161. St-Hilaire M. Topological Planning and Design of UMTS Mobile Networks: a Survey. // Wireless Communications and Mobile Computing. 2009. Vol. 9 (7) – pp. 948-958.

162. Stützle T., Hoos H.H. MAX–MIN Ant System. // Future Generation Computer Systems. 2000. Vol. 16 (8) – pp. 889-914.

163. Szeto W.Y., Wu Y., Ho S.C. An Artificial Bee Colony Algorithm for the Capacitated Vehicle Routing Problem. // European Journal of Operational Research. 2011. Vol. 215 (1) – pp. 126-135.

164. Szlovencsak A., Godor I., Harmatos J., Cinkler T. Planning Reliable UMTS Terrestrial Access Networks. // IEEE Communications Magazine. 2002. Vol. 40 (1) – pp. 66-72.

165. Talbi E.G. Metaheuristics: From Design to Implementation. – Wiley, 2009. – 624 p.

166. Teodorović D., Šelmić M., Davidović T. Bee Colony Optimization Part I: The Algorithm Overview. // Yugoslav Journal of Operations Research. 2015. Vol. 25 (1) – pp. 33-56.

167. Teodorović D., Šelmić M., Davidović T. Bee Colony Optimization Part II: The Application Survey. // Yugoslav Journal of Operations Research. 2015. Vol. 25 (2) – pp. 185-219.

168. Ting C.J., Chen C.H. A Multiple Ant Colony Optimization Algorithm for the Capacitated Location Routing Problem. // International Journal of Production Economics. 2013. Vol. 141 (1) – pp. 34-44.

169. Tsagkaris K., Demestichas P. WiMax Network. // IEEE Vehicular Technology Magazine. 2009. Vol. 4 (2) – pp. 24-35.

170. Van Laarhoven P.J.M., Aarts E.H.L., Lenstra J.K. Job Shop Scheduling by Simulated Annealing. // Operations Research. 1992. Vol. 40 (1) – pp. 113-125.

171. Van Roy T.J. A Cross Decomposition Algorithm for Capacitated Facility Location. // Operations Research. 1986. Vol. 34 (1) – pp. 145-163.

172. Weber A. Theory of the Location of Industries. – University of Chicago Press, 1929. – 256 p.

173. Wun M.H., Wong L.P., Khader A.T., Tan T.P. A Bee Colony Optimization with Automated Parameter Tuning for Sequential Ordering Problem. // Proceedings of the 4th World Congress on Information and Communication Technologies (WICT 2014). Malacca, Malaysia. 2014, December. – pp. 314-319.

174. Yao B., Hu P., Zhang M., Wang S. Artificial Bee Colony Algorithm with Scanning Strategy for the Periodic Vehicle Routing Problem. // Simulation. 2013. Vol. 89 (6) – pp. 762-770.

175. Yu Y., Murphy S., Murphy L. Planning Base Station and Relay Station Locations for IEEE 802.16j Network with Capacity Constraints. // Proceedings of the 7th IEEE Consumer Communications and Networking Conference. Las Vegas, USA. 2010, January. – pp. 1-5.

ПРИЛОЖЕНИЕ 1. СИНТАКСИС ПСЕВДОКОДА ПРИВОДИМЫХ В ДИССЕРТАЦИИ АЛГОРИТМОВ

Условный оператор (оператор ветвления)

Обеспечивает выполнение определенного набора команд только при условии истинности некоторого логического выражения.

Условный оператор с одной ветвью

```
IF условие THEN
     команды;
END IF;
```

Подобный оператор начинается с ключевого слова `IF`. При выполнении условного оператора вычисляется `условие`, и если оно истинно, то выполняются команды до ключевых слов `END IF`, иначе выполнение программы продолжается со следующей за условным оператором команды [36].

Условный оператор с двумя ветвями

```
IF условие THEN
     команды_1;
ELSE
     команды_2;
END IF;
```

В данном случае при истинности `условия` выполняются `команды_1`, при ложности – `команды_2`.

Условный оператор с несколькими ветвями

```
IF условие_1 THEN
     команды_1;
ELSE IF условие_2 THEN
     команды_2;
ELSE IF условие_3 THEN
```

```
     команды_3;
...
ELSE IF условие_(N-1) THEN
     команды_(N-1);
ELSE
     команды_N;
END IF;
```

Условный оператор с несколькими ветвями реализует последовательную проверку сразу (N-1) условий. Как только среди них встретится истинное, будет выполнен соответствующий набор команд и исполнение перейдёт к команде, следующей за условным оператором. Если ни одно из условий не является истинным, то выполнятся `команды_N` из ветви `ELSE`.

Цикл

Обеспечивает многократное выполнение определенного набора команд.

Цикл с предусловием

```
WHILE условие DO
     тело цикла;
END WHILE;
```

Цикл начинается с оператора `WHILE`. Данный цикл выполняется пока истинно некоторое `условие`, указанное перед его началом [68]. Так как `условие` проверяется до выполнения `тела цикла`, тело может быть не выполнено ни разу.

Цикл с постусловием

```
REPEAT
     тело цикла;
UNTIL условие;
```

Цикл начинается с оператора `REPEAT`. Данный цикл выполнится, по крайней мере, один раз, так как `условие` проверяется после выполнения `тела цикла`.

Цикл со счетчиком

```
FOR c = b TO e BY s DO
     тело цикла;
END FOR;
```

В цикле со счетчиком котором некоторая переменная (счетчик `c`) меняет своё значение от заданного начального значения (`b`) до конечного значения (`e`) с некоторым шагом (`s`), причем для каждого значения счетчика тело цикла выполняется один раз [68]. Подобный цикл начинается с оператора `FOR`. Если в объявлении цикла отсутствует описание шага (`BY s`), подразумевается, что шаг равен единице.

Совместный цикл

```
FOREACH item in set DO
     тело цикла (использование item);
END FOREACH;
```

Подобный цикл задает выполнение некоторых команд для объектов `item` из заданного множества `set`, без явного указания порядка перечисления этих объектов. Является формальной записью словесной инструкции вида «Выполнить тело цикла для каждого элемента множества `set`».

Выход из цикла

```
WHILE условие DO
     часть тела цикла;
     IF условие выхода THEN
          BREAK;
     END IF;
     часть тела цикла;
END WHILE;
```

Выход из цикла осуществляется при помощи ключевого слова `BREAK`. При выполнении некоторого `условия выхода` выполнение программы продолжается со следующей за оператором цикла команды.

Досрочный переход на следующую итерацию цикла

```
WHILE условие DO
     начальная часть тела цикла;
     IF условие перехода THEN
          CONTINUE;
     END IF;
     конечная часть тела цикла;
END WHILE;
```

Выход из текущего витка цикла (с переходом на следующую итерацию цикла) при помощи ключевого слова `CONTINUE`. При выполнении некоторого `условия перехода` не будет выполнена `конечная часть тела цикла`, а выполнение программы будет передано на следующую итерацию цикла (вернее на проверку `условия`).

Отметим, что команды `BREAK` и `CONTINUE` работают не только в циклах `WHILE`, но и в циклах, определяемых другими ключевыми словами (`FOR`, `FOREACH`, `REPEAT`).

Другие команды

Оператор return

```
RETURN возвращаемое_значение;
```

Подобный оператор обозначает завершение алгоритма (или части алгоритма, например, функции). `Возвращаемое_значение` представляет собой результат работы алгоритма.

Оператор инкрементирования

```
x++;
```

Подобным образом обозначается увеличение переменной `x` на единицу.

Оператор декрементирования

```
x--;
```

Подобным образом обозначается уменьшение переменной `x` на единицу.

Остаток от деления

```
x MOD y;
```

Подобным образом обозначается операция получения остатка от деления `x` на `y`.

Генерация случайного числа с плавающей запятой

```
x = Random;
```

Функция `Random` генерирует случайные числа с плавающей запятой в диапазоне [0; 1].

Генерация случайного целого числа

```
x = Rand(y);
```

Функция `Rand(y)` генерирует целые случайные числа из диапазона 0, 1, 2, …, (`y-1`).

Комментарии

Комментарии представляют собой пояснения к псевдокоду, располагаемые непосредственно внутри комментируемого текста программы. Комментарии не оказывают никакого влияния на результат выполнения кода алгоритма, т.к. не являются командами.

Однострочный комментарий

```
команда_1; // текст однострочного комментария
команда_2;
// текст однострочного комментария
команда_3;
```

Однострочный комментарий может располагаться как на отдельной строке, так и на одной строке с командой. Все, что написано после символов //, считается комментарием и не влияет на семантику команд.

Многострочный комментарий

```
команда_1; /* текст многострочного комментария
текст многострочного комментария
…
текст многострочного комментария */
команда_2;
/* текст многострочного комментария
текст многострочного комментария
…
текст многострочного комментария */
команда_3;
```

Общие соглашения о синтаксисе

Оператор присваивания

```
A = B;
```

Проверка условия равенства

```
A == B;
```

Проверка условия неравенства

```
A != B;
```

Логическое отрицание

```
NOT A;
```

Логическое И

```
A AND B;
```

Логическое ИЛИ

```
A OR B;
```

Нумерация строк

В тексте диссертации псевдокод большинства приводимых алгоритмов представлен в виде последовательности пронумерованных команд. Команды пронумерованы для удобства ссылки на них из текста работы и улучшения читаемости псевдокода.

```
1:   команда_1;
2:   команда_2;
3:   команда_3;
     …
n:   команда_n;
```

ПРИЛОЖЕНИЕ 2. ПСЕВДОКОД И БЛОК-СХЕМЫ НЕКОТОРЫХ АЛГОРИТМОВ

Процедура скрещивания для ЭА решения ЗРЭРИС

Используется *n_points*-точечное скрещивание. На вход алгоритма подается целочисленный массив *Points* размерности *n_points*, элементы которого – позиции точек скрещивания. Массив создается следующим образом: необходимо сгенерировать *n_points* разных целых чисел из диапазона $[1; N_{ps} - 1]$, а затем отсортировать их по возрастанию.

Отметим, что результатом работы алгоритма будет ∅, если получившееся решение не является допустимым.

Алгоритм П2.1 Псевдокод процедуры скрещивания для ЗРЭРИС

```
/* Введем обозначения: X1 и Y1 – хромосомы 1-го родителя, выбранного для
   скрещивания; X2 и Y2 – хромосомы 2-го родителя, выбранного для
   скрещивания; Xchild и Ychild – хромосомы потомка; num – номер (индекс)
   текущей точки скрещивания в массиве Points.  */
1:  Заполнить массив Ychild нулями;
2:  num = 1;
3:  FOR i = 1 TO Ntp DO    // создание хромосомы Xchild
4:      IF (i > Points[num]) THEN
5:          num++;
6:      END IF;
    /* Потомок берет от 1-го родителя часть хромосомы X, расположенную
       между четной и нечетной точкой скрещивания, от 2-го родителя – часть
       хромосомы X, расположенную между нечетной и четной точкой  */
7:      IF (num MOD 2) != 0 THEN
8:          Xchild[i] = X1[i];
9:      ELSE
```

```
        X_child[i] = X_2[i];
    END IF;
    Y_child[X_child[i]] = ∞;   // пометим ненулевые элементы Y_child
END FOR;
FOR s = 1 TO N_ps DO   // заполнение хромосомы Y_child
    IF (Y_child[s] = ∞) THEN
        IF (Random < 0.5) THEN
            Y_child[s] = Y_1[s];
        ELSE
            Y_child[s] = Y_2[s];
        END IF;
    END IF;
END FOR;
IF (потомок удовлетворяет ограничениям (1.18) и (1.19)) THEN
    RETURN (X_child и Y_child);
ELSE
    RETURN ∅;
END IF;
```

Пчелиный алгоритм BCOi для задачи минимизации

Алгоритм П2.2 Пчелиный алгоритм BCOi

```
    /* Введем обозначения: x_best – лучшее решение задачи.  */
1:  FOR b = 1 TO B DO
2:      Поставить пчеле b в соответствие решение x_b;
3:  END FOR;
4:  FOR s = 1 TO NC DO
5:      FOR b = 1 TO B DO            // прямой проход
6:          Построить окрестность соседних решений N(x_b);
7:          Выбрать согласно методу рулетки одно решение из N(x_b),
            обозначим его z_b;
8:      END FOR;
9:      Найти минимальное и максимальное значение целевой функции
        в пределах улья;             // начало обратного прохода
10:     FOR b = 1 TO B DO
11:         Рассчитать значение O_b по формуле (2.9);
12:     END FOR;
13:     Найти максимальное значение нормализованной целевой
        функции в пределах улья;
14:     FOR b = 1 TO B DO
15:         Для пчелы b принять решение о лояльности текущему
            решению согласно формуле (2.8);
16:     END FOR;
17:     FOR b = 1 TO B DO
18:         Если пчела b отказалась от своего решения, то при
            помощи метода рулетки (по формуле (2.10)) выбрать для
            нее рекрута;
```

```
19:     END FOR;
20: END FOR;
21: Найдем x_NC – лучшее решение в улье;
22: IF (x_NC лучше x_best) THEN
23:     x_best = x_NC;
24: END IF;
24  IF (условие останова не выполнено) THEN
26:     переход к команде 1;
27: END IF;
28: RETURN x_best;
```

Пчелиный алгоритм ABC для задачи минимизации

Алгоритм П2.3 Пчелиный алгоритм ABC

/* Введем обозначения: *n_iter* – массив, содержащий число итераций подряд без улучшения целевой функции для каждого решения; x_{best} – лучшее решение задачи. */

```
1:  FOR i = 1 TO SN DO
2:      n_iter_i = 0;
3:      Инициализировать решение x_i;
4:      Найти fit(x_i);
5:  END FOR;
6:  REPEAT
7:      FOR i = 1 TO SN DO        // действия занятых фуражиров
8:          Найти новое решение v_i по формуле (2.13);
9:          Найти fit(v_i);
10:         IF (fit(v_i) < fit(x_i)) THEN
11:             x_i = v_i;
12:             n_iter_i = 0;
13:         END IF;
14:     END FOR;
15:     Найти вероятности p_i (i = 1, 2, …, SN) по формуле (2.11);
16:     FOR i = 1 TO SN DO        // действия пчел-наблюдателей
17:         Выбрать одно из решений согласно вероятностям p_j
            (j = 1, 2, …, SN), пусть это будет решение x_z;
18:         Найти новое решение v_z по формуле (2.13);
19:         Найти fit(v_z);
20:         IF (fit(v_z) < fit(x_z)) THEN
21:             x_z = v_z;
22:             n_iter_z = 0;
```

```
23:         END IF;
24:     END FOR;
25:     FOR i = 1 TO SN DO        // действия пчел-разведчиков
26:         IF (n_iter_i = limit) THEN
27:             n_iter_i = 0;
28:             Обнулить решение x_i;
29:             Инициализировать решение x_i;
30:         END IF;
31:     END FOR;
32:     Найдем x_CURR_BEST – лучшее решение в улье;
33:     IF (fit(x_CURR_BEST) < fit(x_best)) THEN
34:         x_best = x_CURR_BEST;
35:     END IF;
36: UNTIL (условие останова не выполнено) ;
37: RETURN x_best;
```

Пчелиный алгоритм BCOi для решения задачи размещения элементов развивающихся информационных систем

Алгоритм П2.4 Пчелиный алгоритм BCOi для решения ЗРЭРИС

```
    /* Введем обозначения: Best – лучшее решение задачи; cost_best –
    целевая функция Best.  */
1:  Best = ∅;
2:  cost_best = ∞;
3:  IF (use_global != true) or (Best = ∅) THEN
4:      FOR b = 1 TO B DO
5:          Построить начальное решение Curr_b;
6:      END FOR;
7:  ELSE
8:      Curr_1 = Best;
9:      FOR b = 2 TO B DO
10:         Построить начальное решение Curr_b;
11:     END FOR;
12: END IF;
13: FOR s = 1 TO NC DO
14:     FOR b = 1 TO B DO     // прямой проход
15:         Построим N_better(Curr_b) окрестность решения Curr_b;
16:         W_b = ∅;
17:         FOREACH Sol in N(Curr_b)DO
18:             IF (Sol – недопустимое решение) THEN
19:                 CONTINUE;
20:             END IF;
21:             IF F(Sol) ≥ F(Curr_b) THEN
22:                 CONTINUE;
```

```
23:         END IF;
24:         Добавить Sol в множество Wb;
25:     END FOREACH;
26:     IF (Wb != ∅) THEN
27:         Выберем одно решение из Wb при помощи одного из
            методов (пропорциональный, турнирный, ранговый,
            дизруптивный) и присвоим его Currb;
28:     END IF;
29:     IF (F(Currb) < cost_best) THEN
30:         Best = Currb;
31:         cost_best = F(Currb);
32:     END IF;
33: END FOR;
34: // обратный проход
35: R = 0;
36: Gnon_loyal = ∅;     /* множество пчел, которые отказались от своих
    текущих решений */
37: Gloyal = ∅;     /* множество пчел, которые не отказались от своих
    текущих решений */
38: IF (s != NC) THEN
39:     Рассчитать нормализованные значения целевой функции
        для каждой пчелы по формуле (2.9);
40:     FOR b = 1 TO B DO
41:         Найти p_b^{u+1} по формуле (2.8);
42:         z = Random;
43:         IF (z > p_b^{u+1}) THEN
44:             Добавим пчелу b в множество Gnon_loyal;
```

```
45:                 ELSE
46:                     R++;
47:                     Добавим пчелу b в множество G_loyal;
48:                 END IF;
49:             END FOR;
50:             FOR b = 1 TO B DO
51:                 IF (b ∈ G_non_loyal) THEN
52:                     При помощи одного из методов
                        (пропорциональный, турнирный, ранговый,
                        дизруптивный) присвоим пчеле b решение одной
                        из пчел из множества G_loyal;
53:                 END IF;
54:             END FOR;
55:         END IF;
56:     END FOR;
57:     Обнулим все решения Curr_b (b = 1, 2, …, B);
58:     IF (время работы алгоритма time не истекло) THEN
59:         переход к команде 3;
60:     END IF;
61:     RETURN Best;
```

Пчелиный алгоритм ABC для решения задачи размещения элементов развивающихся информационных систем

Алгоритм П2.5 Пчелиный алгоритм ABC для решения ЗРЭРИС

/* Введем обозначения: *Best* – лучшее решение задачи; *cost_best* – целевая функция *Best*; *n_iter* – массив, содержащий число итераций подряд без улучшения целевой функции для каждого решения */

1: *Best* = ∅;
2: *cost_best* = ∞;
3: FOR *i* = 1 TO *SN* DO
4: *n_iter*$_i$ = 0;
5: Построить начальное решение *Curr*$_i$;
6: END FOR;
7: REPEAT
8: FOR *i* = 1 TO *SN* DO // действия занятых фуражиров
9: Построим *Sol* – случайное допустимое решение из окрестности N_{better}(*Curr*$_i$);
10: IF (*F(Sol)* < *F(Curr*$_i$*)*) THEN
11: *n_iter*$_i$ = 0;
12: *Curr*$_i$ = *Sol*;
13: END IF;
14: END FOR;
15: Согласно одному из методов (пропорциональный, турнирный, ранговый, дизруптивный) для каждого решения *Curr*$_i$ (*i* = 1, 2, …, *SN*) найдем p_i – вероятность того, что оно будет выбрано пчелой-наблюдателем;
16: FOR *i* = 1 TO *SN* DO // действия пчел-наблюдателей
17: Выбрать одно из решений согласно вероятностям p_j (*j* = 1, 2, …, *SN*), пусть это будет решение *Curr*$_z$;
18: Построим *Sol* – случайное допустимое решение из

```
        окрестности N_better(Curr_z);
19:     IF (F(Sol) < F(Curr_z)) THEN
20:         n_iter_z = 0;
21:         Curr_z = Sol;
22:     END IF;
23: END FOR;

24: FOR i = 1 TO SN DO        // обновление лучшего решения
25:     IF (F(Curr_i) < cost_best) THEN
26:         Best = Curr_i;
27:         cost_best = Curr_i;
28:     END IF;
29: END FOR;

30: FOR i = 1 TO SN DO        // действия пчел-разведчиков
31:     IF (n_iter_i = limit) THEN
32:         n_iter_i = 0;
33:         Curr_i = ∅;
34:         Построить начальное решение Curr_i;
35:     END IF;
36: END FOR;
37: UNTIL (условие останова не выполнено) ;
38: RETURN Best;
```

Алгоритм вероятностного поиска с запретами для решения ЗРЭРИС

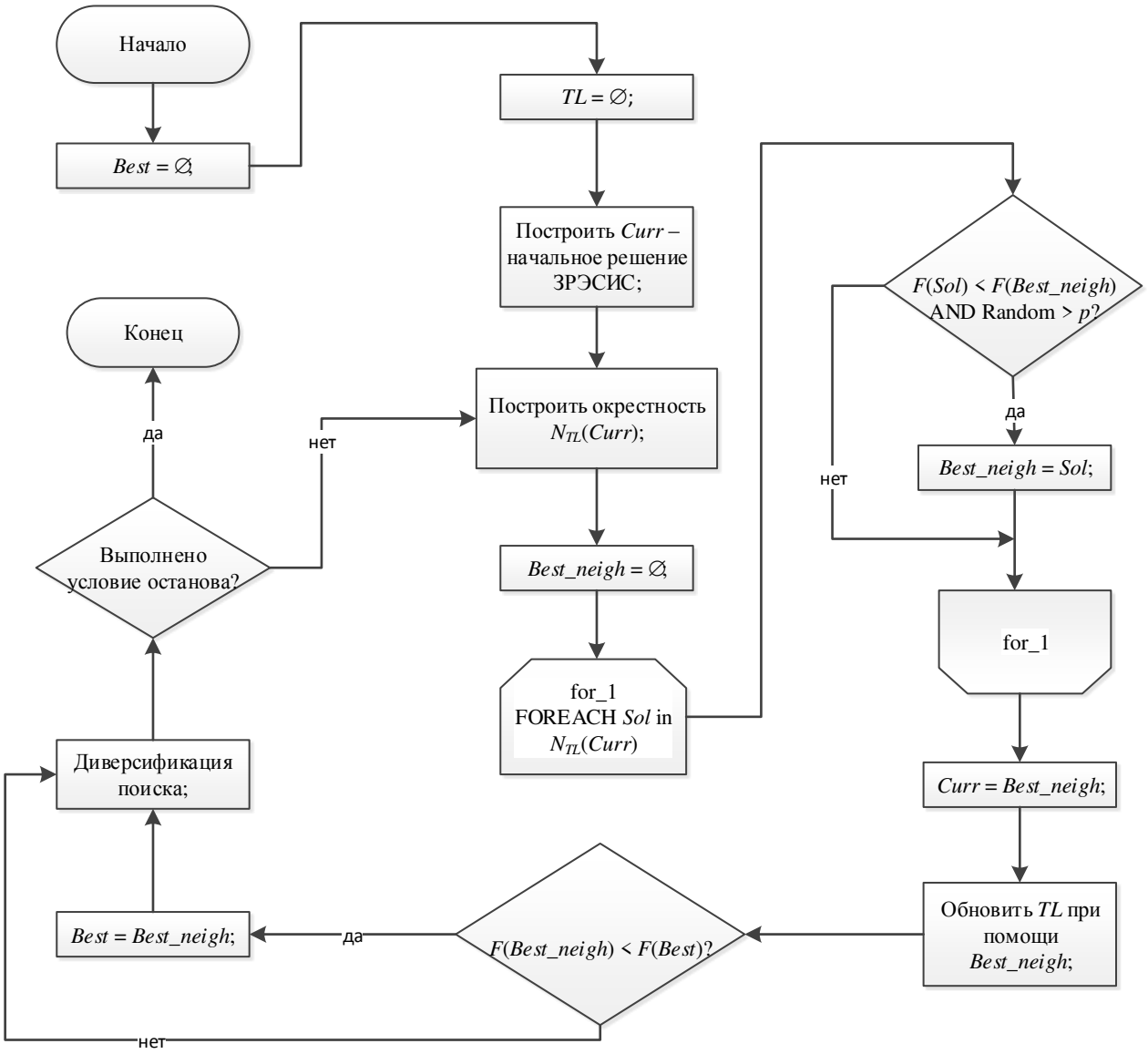

Рисунок П2.1 – Блок-схема вероятностного ПЗ для решения ЗРЭРИС

Алгоритм мультистарта для решения ЗРЭРИС

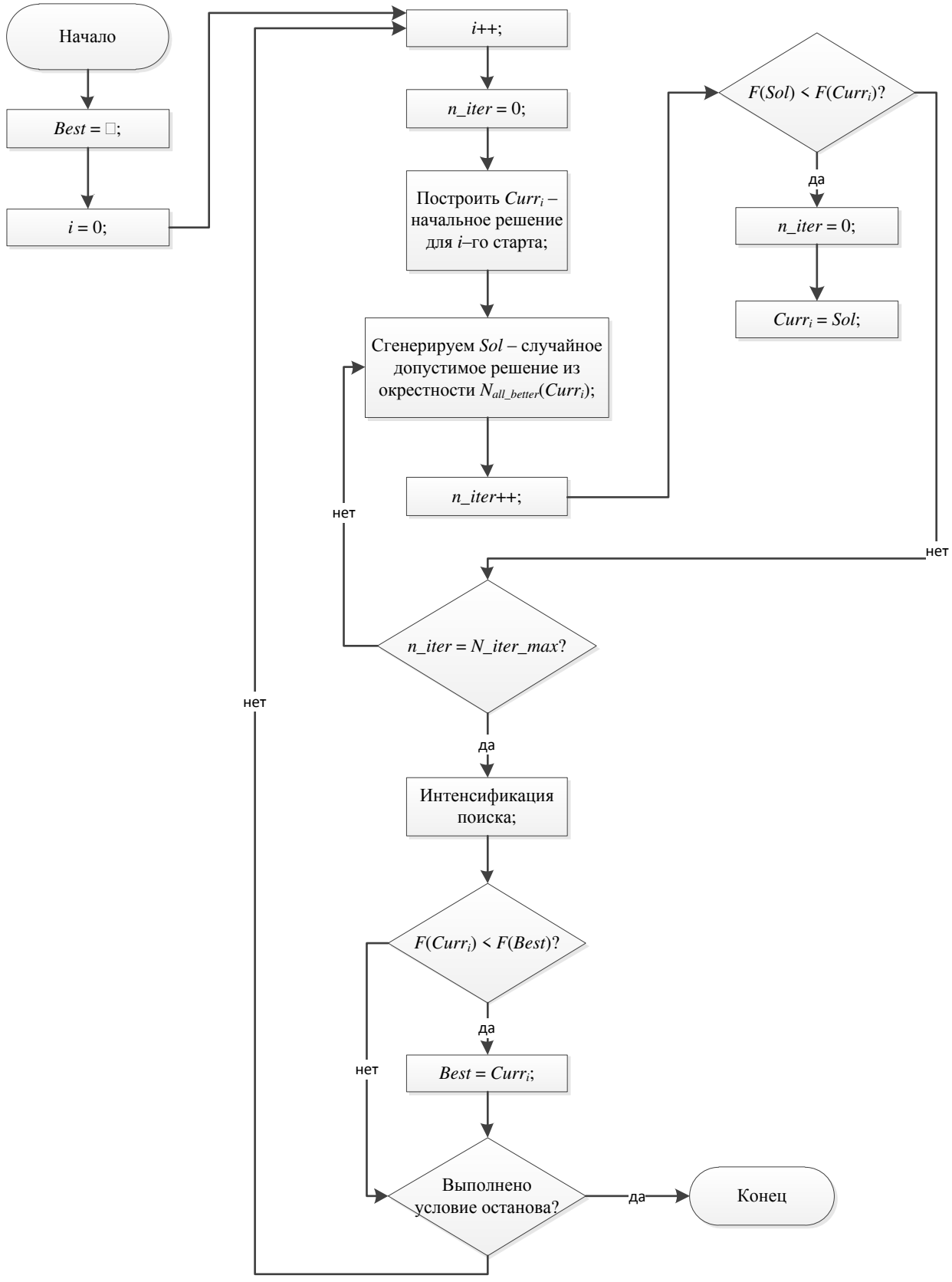

Рисунок П2.2 – Блок-схема алгоритма мультистарта для решения ЗРЭРИС

Муравьиный алгоритм решения ЗРЭРИС

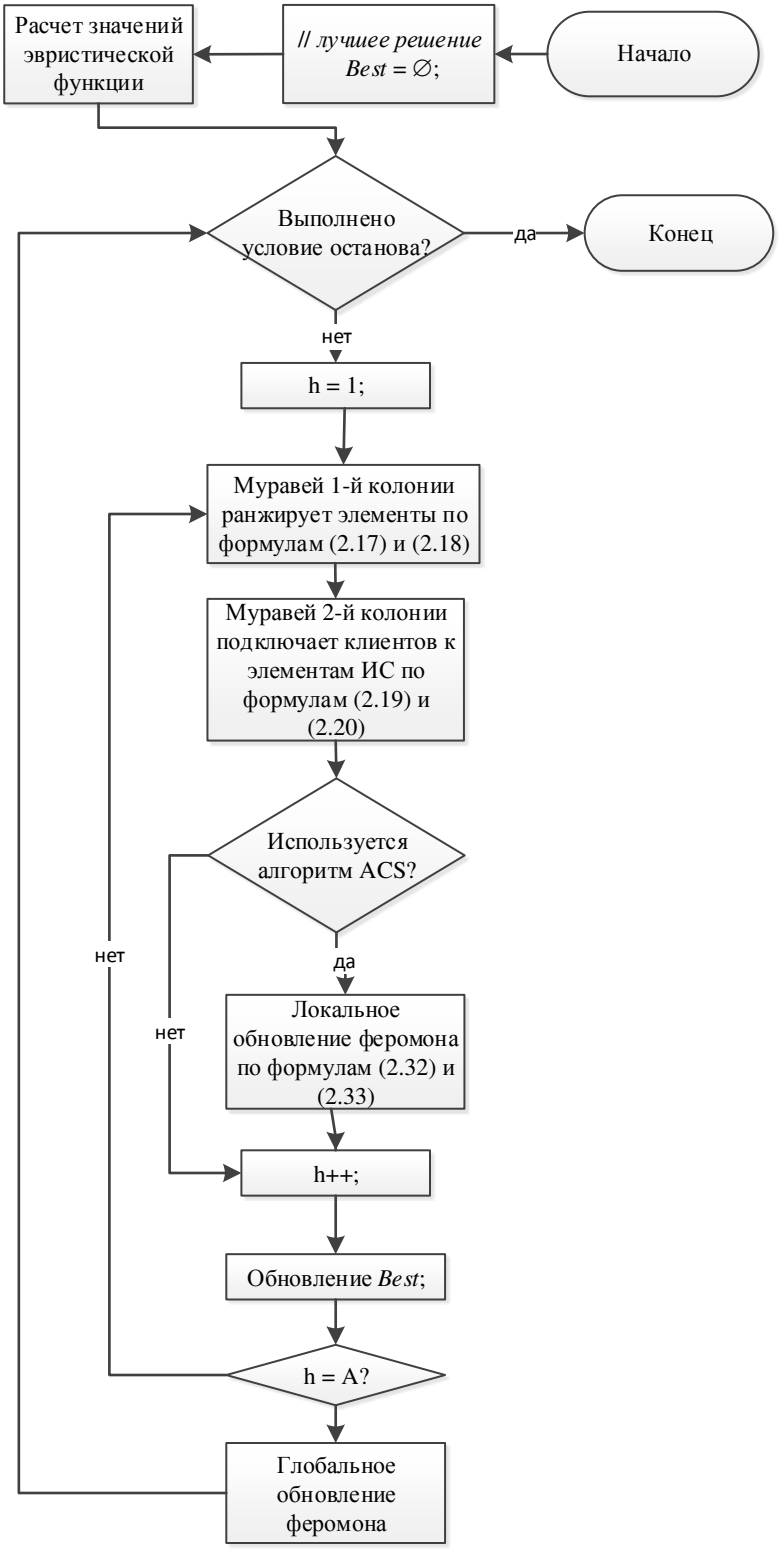

Рисунок П2.3 – Блок-схема муравьиного алгоритма решения ЗРЭРИС

Метод настройки параметров метаэвристик решения ЗРЭРИС

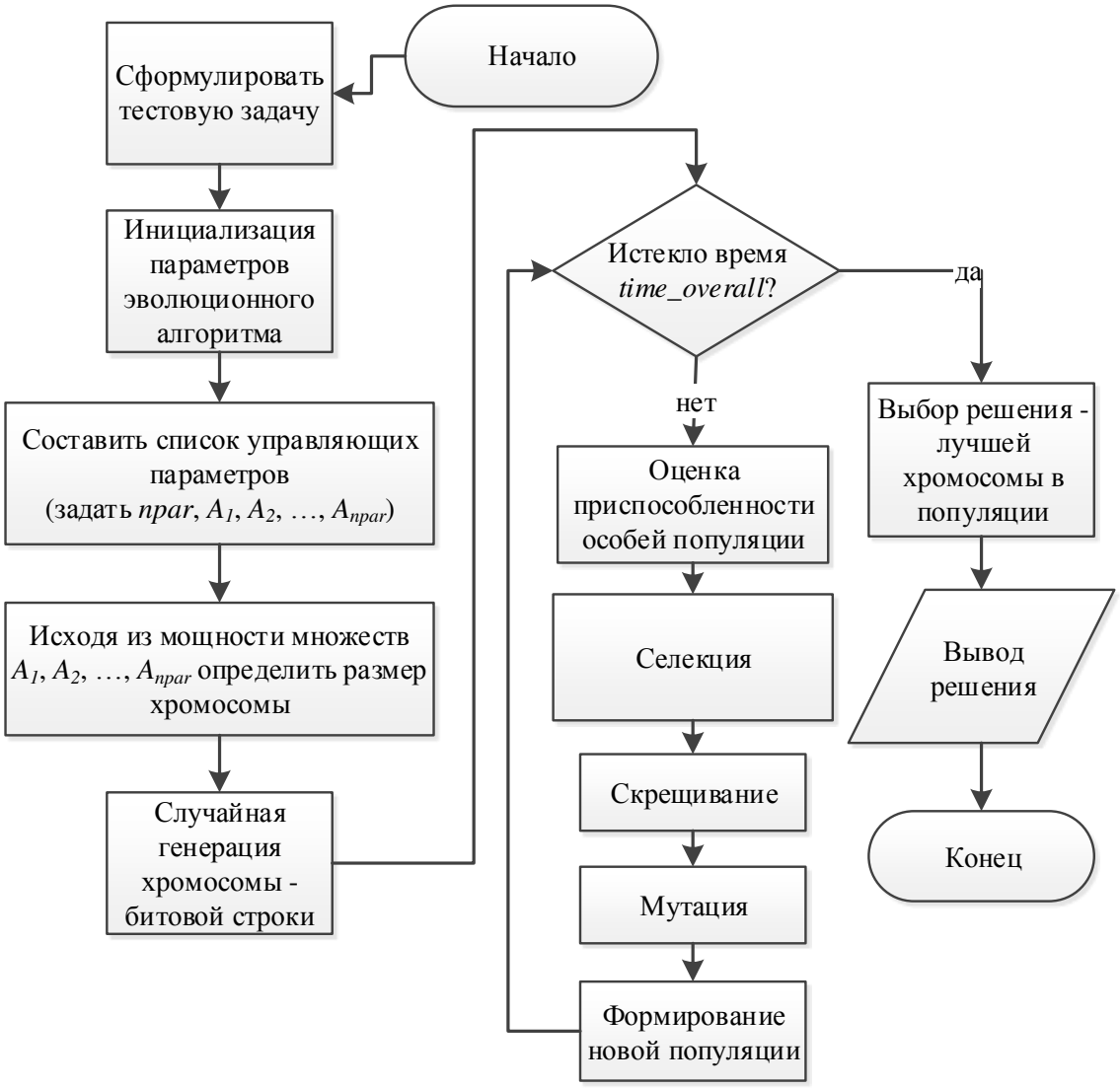

Рисунок П2.4 – Блок-схема первой фазы метода оптимизации параметров метаэвристик решения ЗРЭРИС

ПРИЛОЖЕНИЕ 3. СКРИНШОТЫ НЕКОТОРЫХ ОКОН СОЗДАННОГО ПРОГРАММНОГО КОМПЛЕКСА

Вид главного окна программного комплекса сразу после запуска представлен на рисунке П3.1.

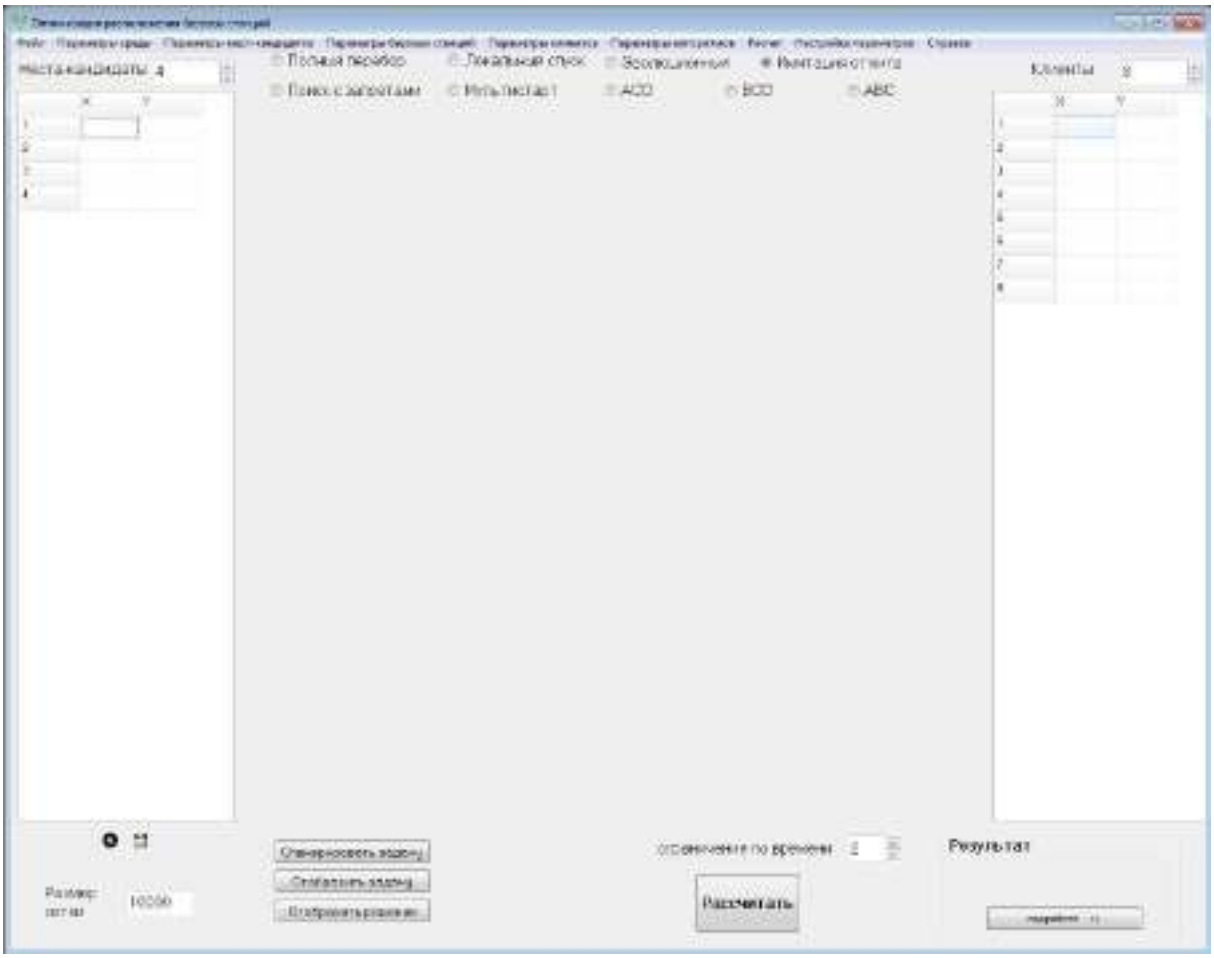

Рисунок П3.1 – Исходный вид главного окна программы

В центре главного окна расположена карта (сетка), на которой размещаются клиенты и места-кандидаты. Имеется возможность менять масштаб карты. На карте клиенты обозначаются зелеными квадратиками, а места-кандидаты – синими кружками. По обеим сторонам от сетки находятся таблицы, в которых отображаются координаты мест-кандидатов и клиентов. Клиенты задаются на карте размещения при помощи клика правой кнопкой мыши, места-кандидаты – при помощи левой. Также имеется возможность задать декартовы координаты объектов непосредственно в таблицах. Вид главной формы программного комплекса с нанесенными на координатную сетку местами-кандидатами и

клиентами (для случая, когда система планируется «с нуля») представлен на рисунке П3.2. Программный комплекс предоставляет возможность загрузки из файла (а также из БД) и записи в файл (а также в БД) основных параметров задачи размещения элементов информационной системы (число и координаты мест-кандидатов, сведения об установленных элементах, число и координаты клиентов). Эта особенность позволяет быстро запускать различные задачи с различными исходными данными. Кроме того, все основные действия, доступные на главном окне с помощью размещенных кнопок, продублированы в меню.

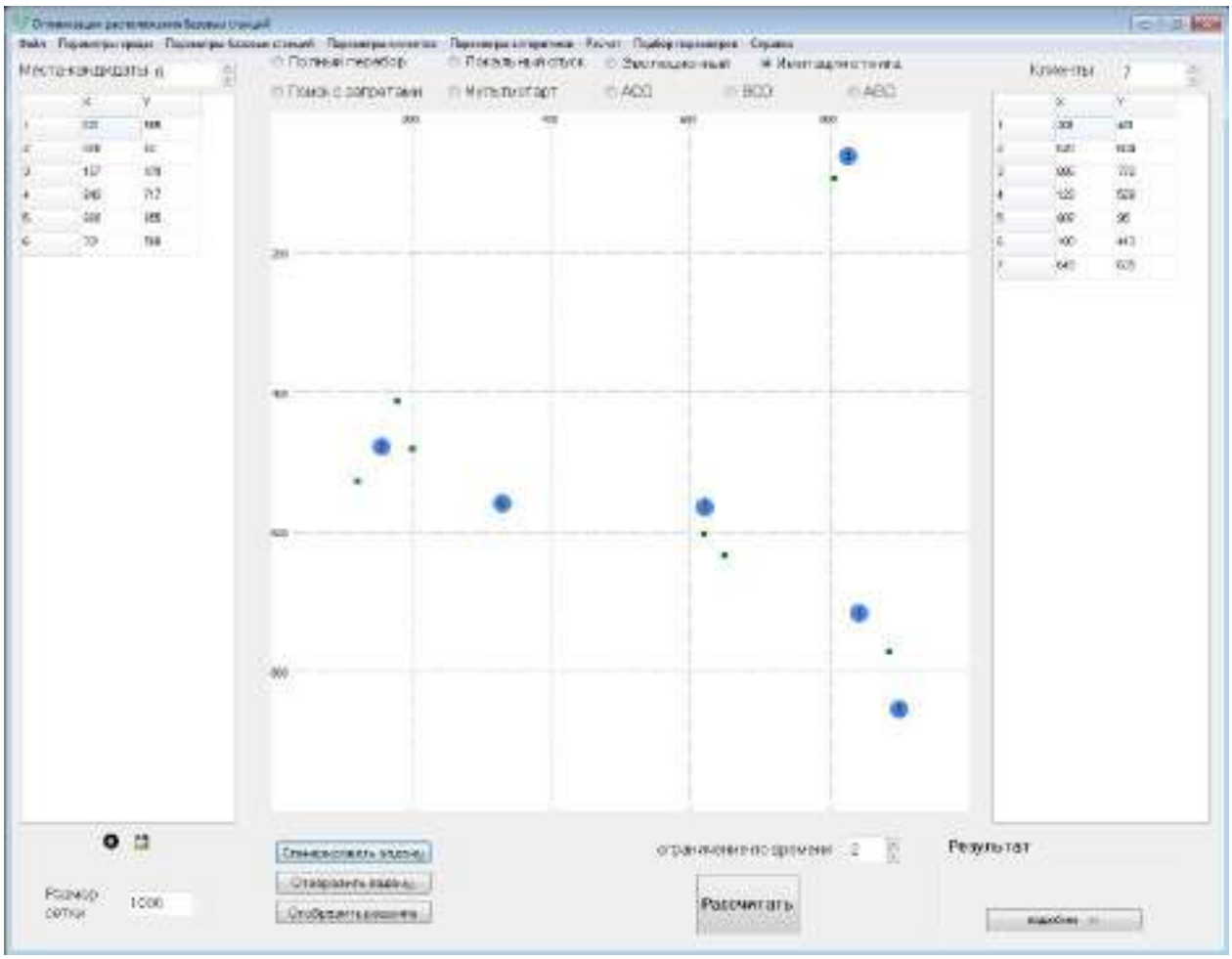

Рисунок П3.2 – Вид главного окна программы с нанесенными местами-кандидатами и клиентами (для проектирования «с нуля»)

Вид главной формы программного комплекса с нанесенными на координатную сетку местами-кандидатами и клиентами для случая модернизации беспроводной сети представлен на рисунке П3.3. Красным цветом обозначено место-кандидат, на котором в исходной ИС уже установлена базовая станция.

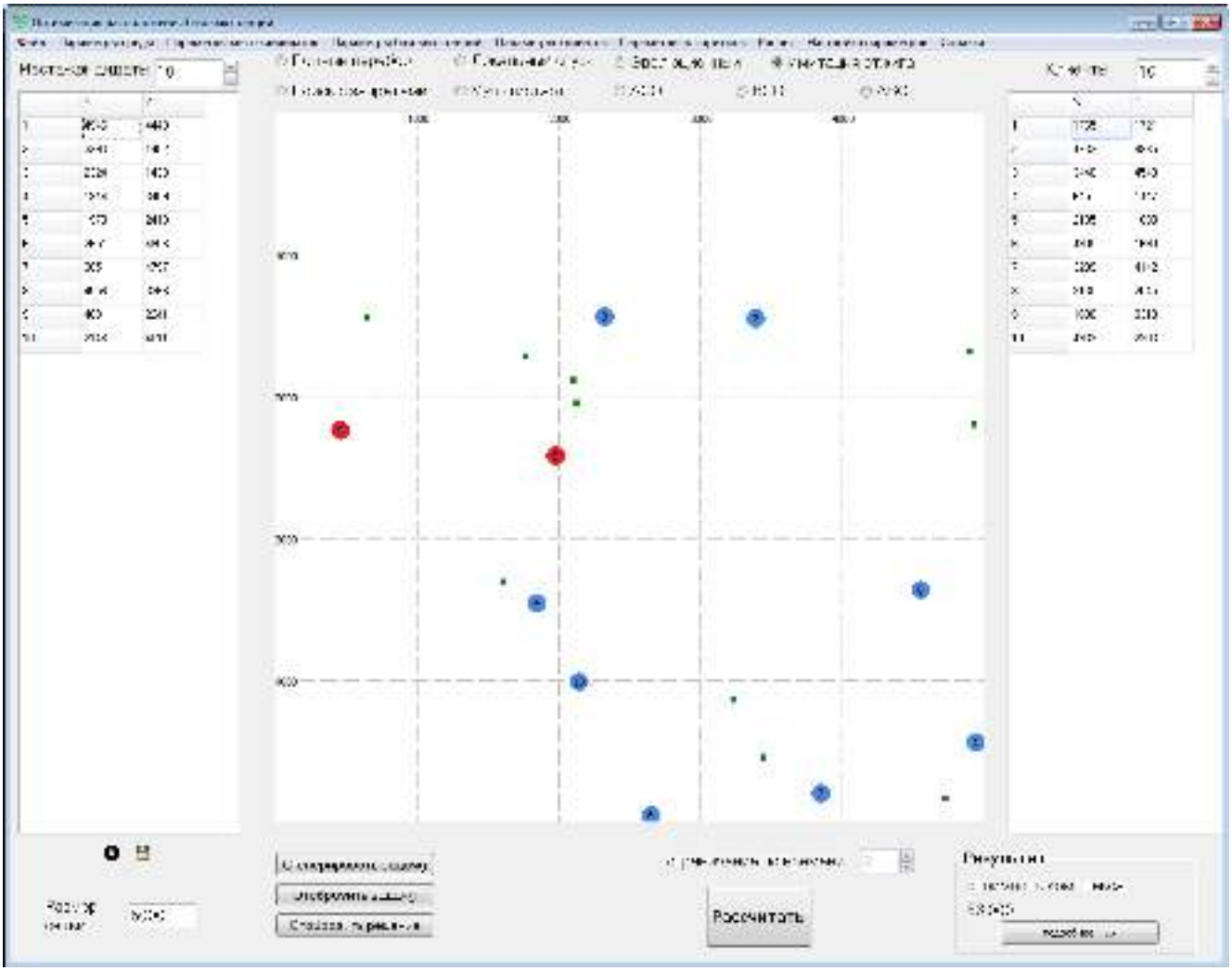

Рисунок П3.3 – Вид главного окна программы с нанесенными местами-кандидатами и клиентами (для случая модернизации ИС)

Задавать настройки для разных типов базовых станций можно на форме «Параметры базовых станций» (рисунок П3.4). Эта форма представляет собой таблицу и дает пользователю возможность изменять такие параметры базовых станций, как цена, производительность, мощность передачи, необходимая мощность принимаемого сигнала, цена удаления из структуры ИС, цена добавления в структуру. Столбец «Использовать в расчетах (+/-)» определяет, будет ли станция определенного типа учитываться при решении задачи. Имеется возможность загрузки из файла (а также из БД) и записи в файл (а также в БД) справочника с параметрами базовых станций.

Задавать настройки для клиентов можно на форме «Параметры клиентов» (рисунок П3.5). Эта форма представляет собой таблицу и дает пользователю

возможность изменять такие параметры клиентов, как требуемая скорость, мощность передачи, необходимая мощность принимаемого сигнала, высота антенны. Имеется возможность загрузки из файла (а также из БД) и записи в файл (а также в БД) настроек клиентов.

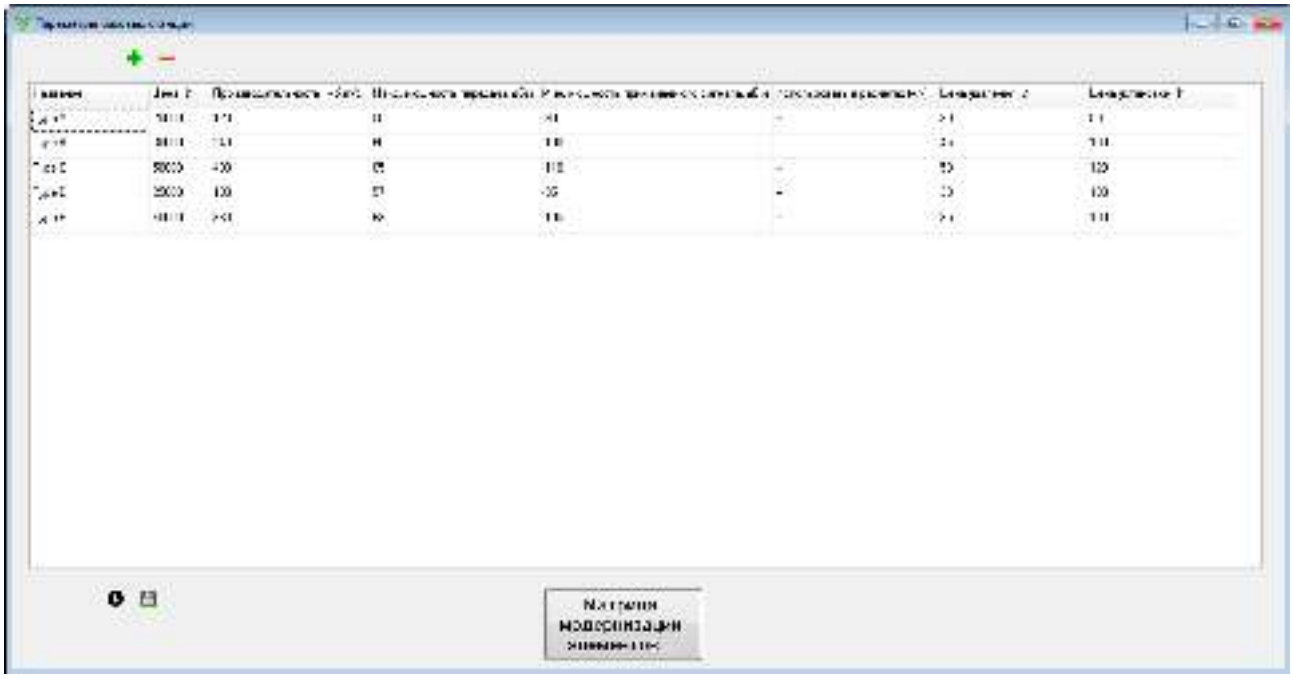

Рисунок П3.4 – Вид формы настройки параметров базовых станций

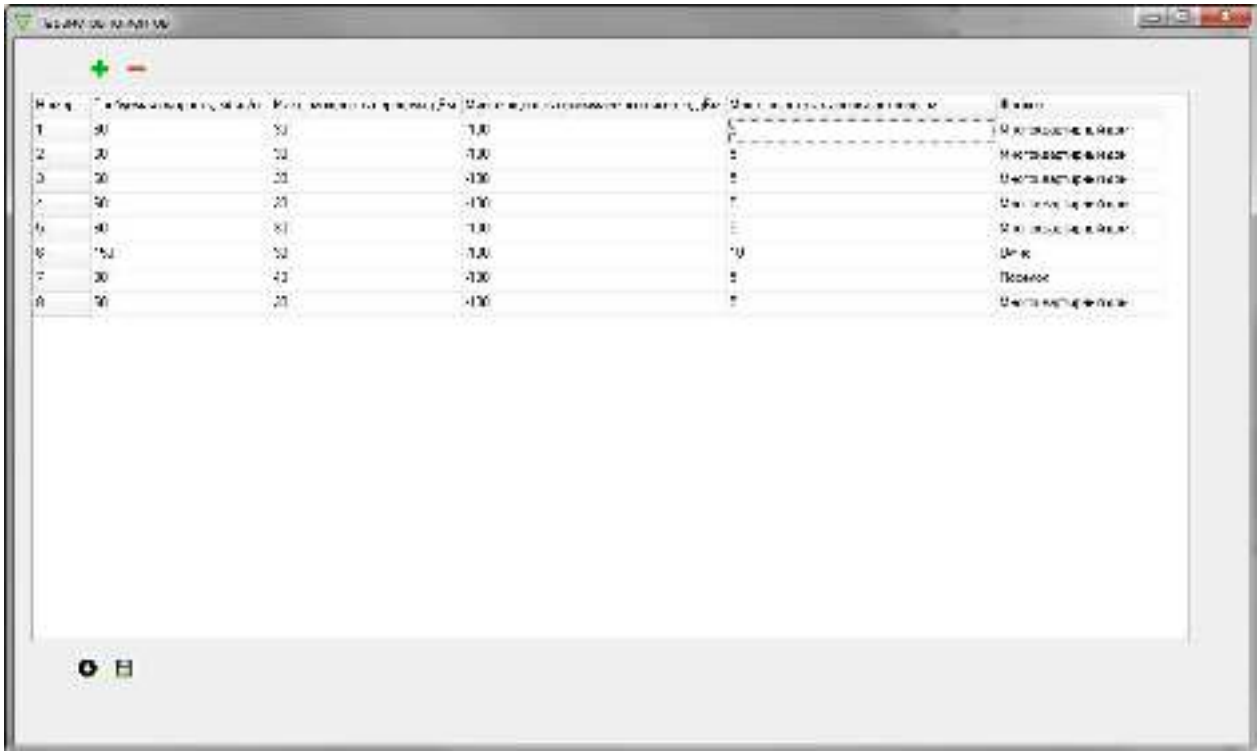

Рисунок П3.5 – Вид формы настройки параметров клиентов

Задавать настройки для мест-кандидатов можно на форме «Параметры мест-кандидатов» (рисунок П3.6). Эта форма представляет собой таблицу и дает пользователю возможность изменять такие параметры мест-кандидатов, как цена аренды, максимальная высота установки антенны, цена удаления элемента ИС, а также содержит информацию о типе установленной на данном месте БС (в случае, если она там установлена). Имеется возможность загрузки из файла (а также из БД) и записи в файл (а также в БД) настроек мест-кандидатов.

Форма «Параметры среды распространения сигнала» (рисунок П3.7) представляет пользователю возможность выбрать из списка модель распространения сигнала, параметры модели, тип местности и т.д.

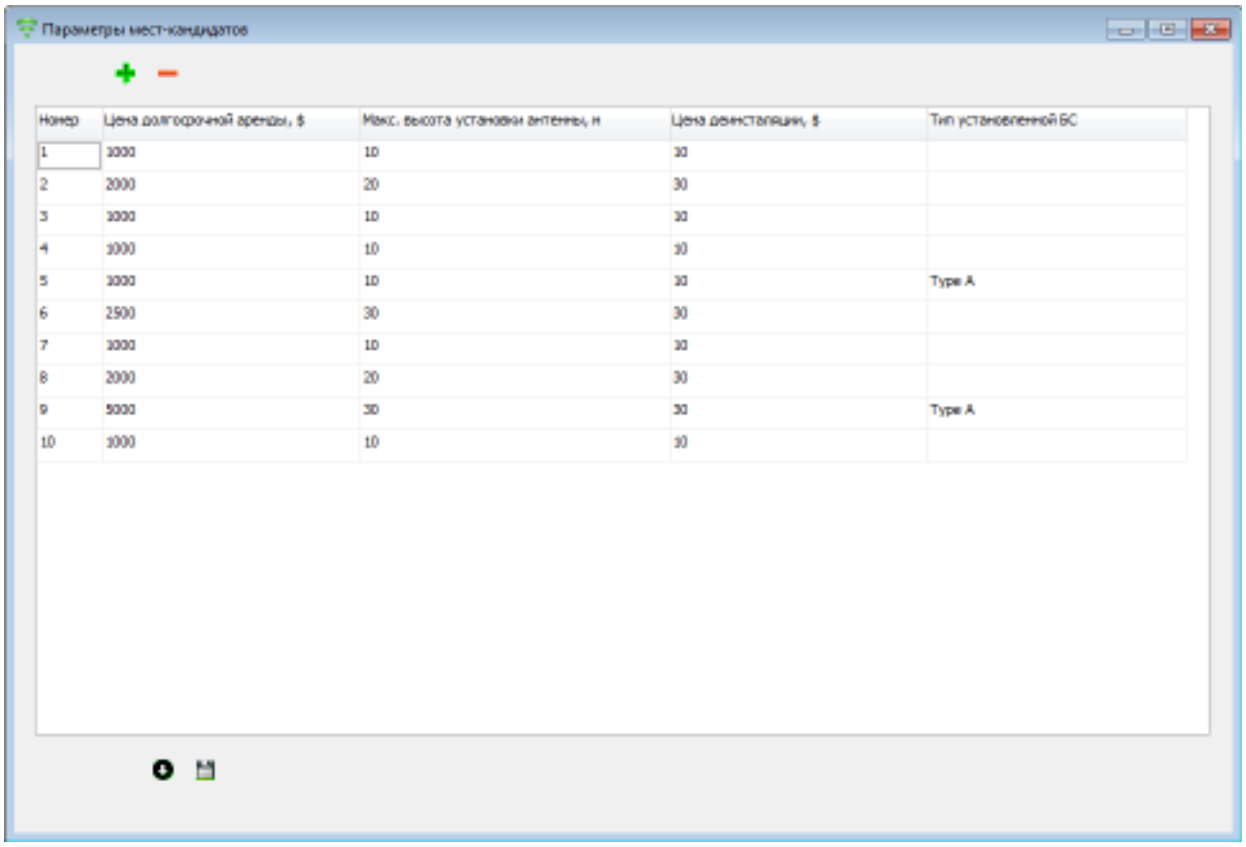

Рисунок П3.6 – Вид формы настройки параметров мест-кандидатов

На главном окне программы осуществляется выбор одного из восьми способов решения задачи: полный перебор, локальный спуск, эволюционный алгоритм, метод имитации отжига, поиск с запретами, алгоритм мультистарта муравьиный алгоритм, пчелиные алгоритмы BCO и ABC. Для изменения параметров настройки алгоритмов решения ЗРЭРИС в меню предусмотрены

кнопки, нажатие на которые приводит к появлению дополнительных окон. Ниже для примера приведен внешний вид формы, предназначенной для ввода параметров муравьиного алгоритма (рисунок П3.8). Для остальных метаэвристик существуют аналогичные формы.

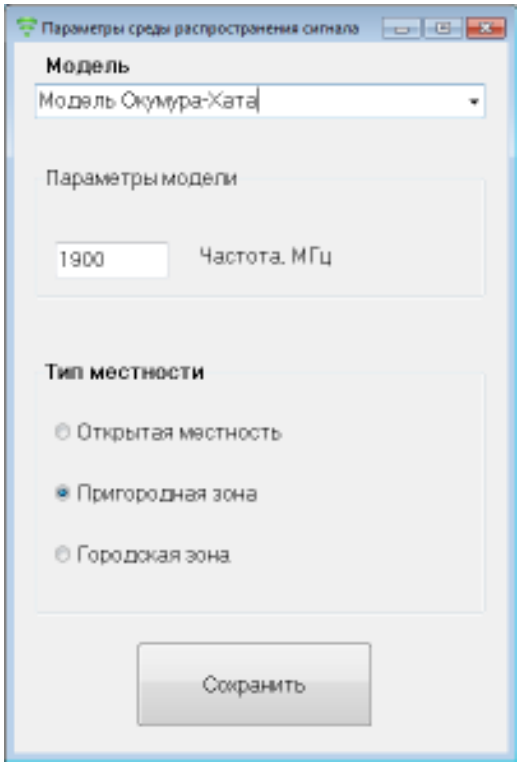

Рисунок П3.7 – Вид формы настройки свойств среды распространения сигнала

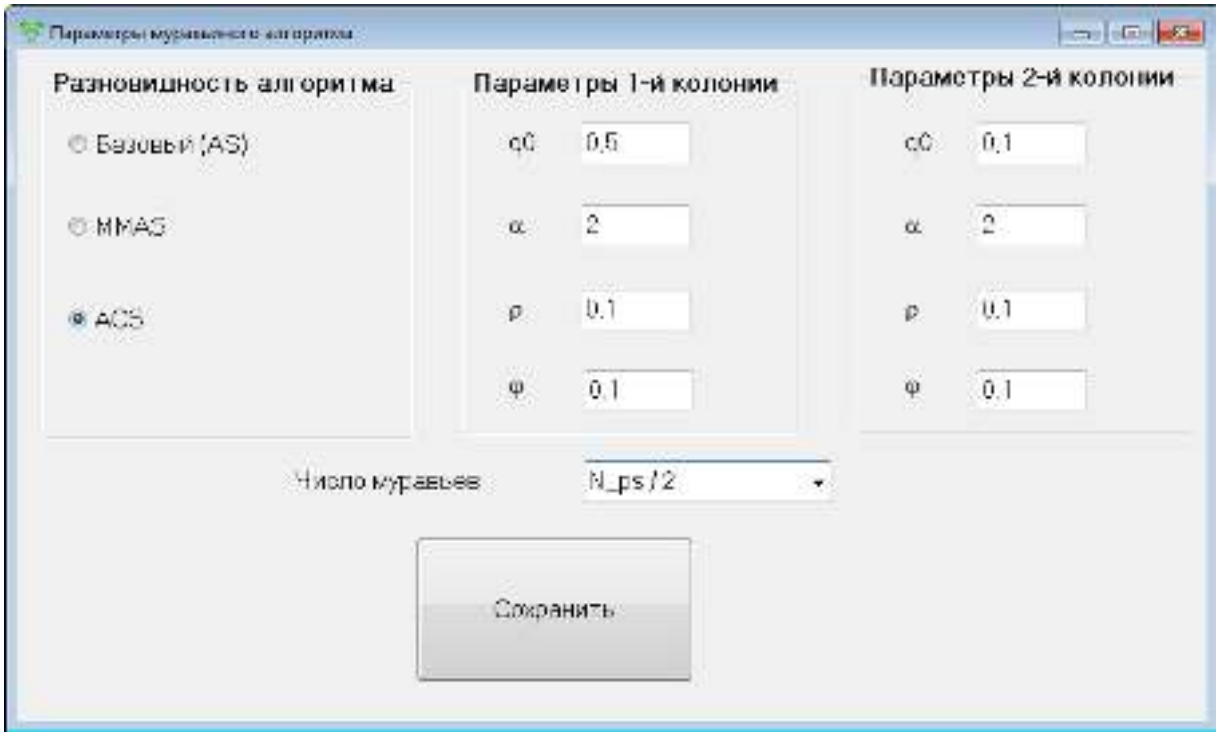

Рисунок П3.8 – Вид формы ввода параметров муравьиного алгоритма

Действия пользователя при работе с созданным программным обеспечением заключаются в следующем. После открытия программы пользователь расставляет на сетке клиенты и места-кандидаты, либо загружает их координаты из файла (из БД). Далее требуется задать параметры БС, клиентов и мест-кандидатов, в т.ч. перечислить БС, которые уже включены в текущую структуру беспроводной сети передачи данных (всю эту информацию можно загрузить из файлов), выбрать способ решения задачи, ввести параметры соответствующего алгоритма и нажать кнопку «Решить».

По окончанию расчетов на координатной сетке главной формы отобразятся результаты расчетов. Места-кандидаты, на которых установлена БС будут обозначены красным цветом, места без базовых станций – светло-розовым (рисунок П3.9). По нажатию на кнопку «Отобразить решение» соединительной линией будет отображен факт подключения клиента к конкретной БС (рисунок П3.10). По нажатию на кнопку «подробнее» откроется форма «Детальные результаты расчетов» (рисунок П3.11). Данное окно содержит информацию о том, на каких местах-кандидатах были установлены БС с указанием конкретного типа БС, а также информацию об удаленных и модернизированных БС. В отдельной таблице для каждого клиента указано, к какой именно базовой станции он подключен. Также на форме «Детальные результаты расчетов» приведена стоимость комплекса базовых станций.

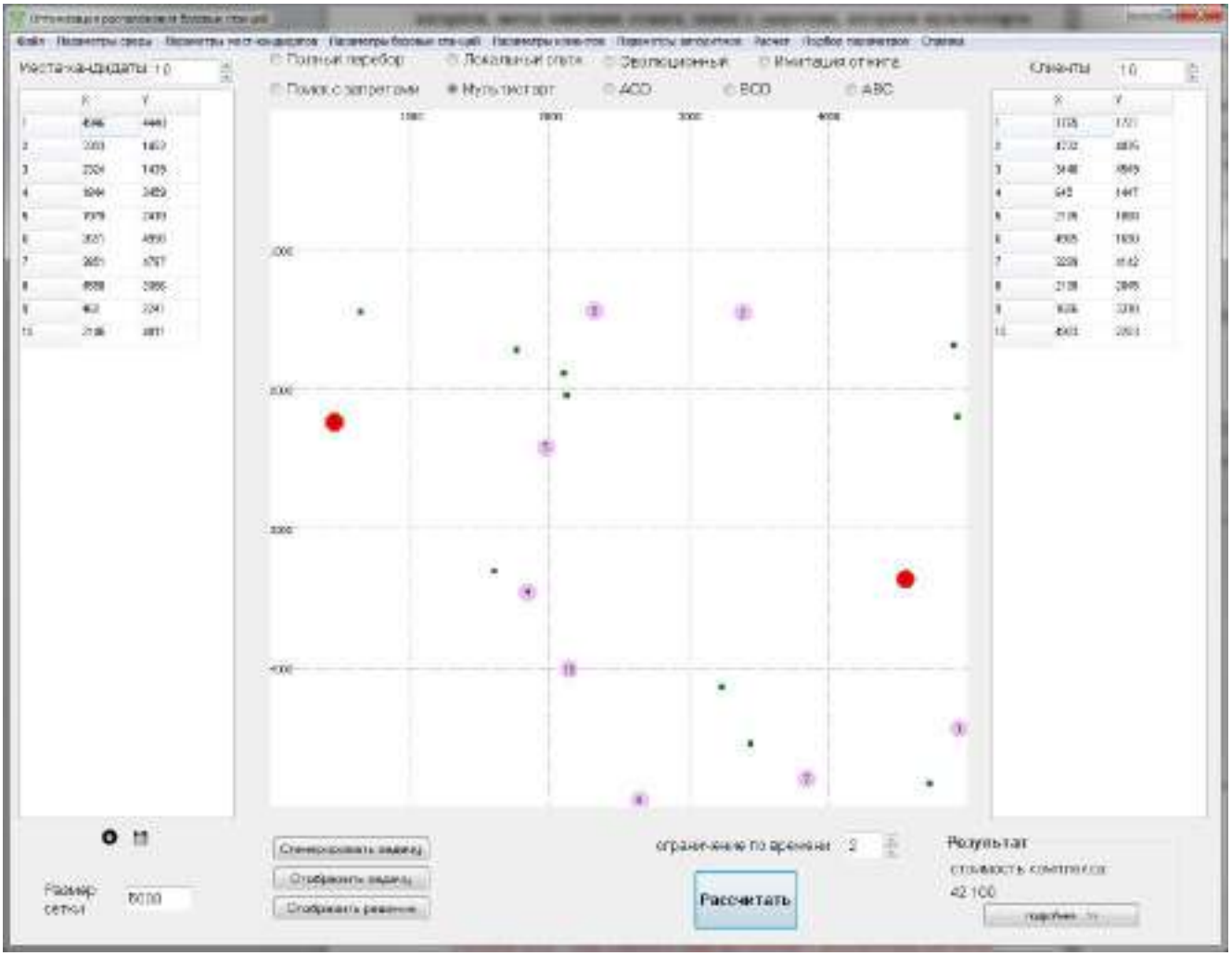

Рисунок П3.9 – Вид главного окна программы с результатами расчетов

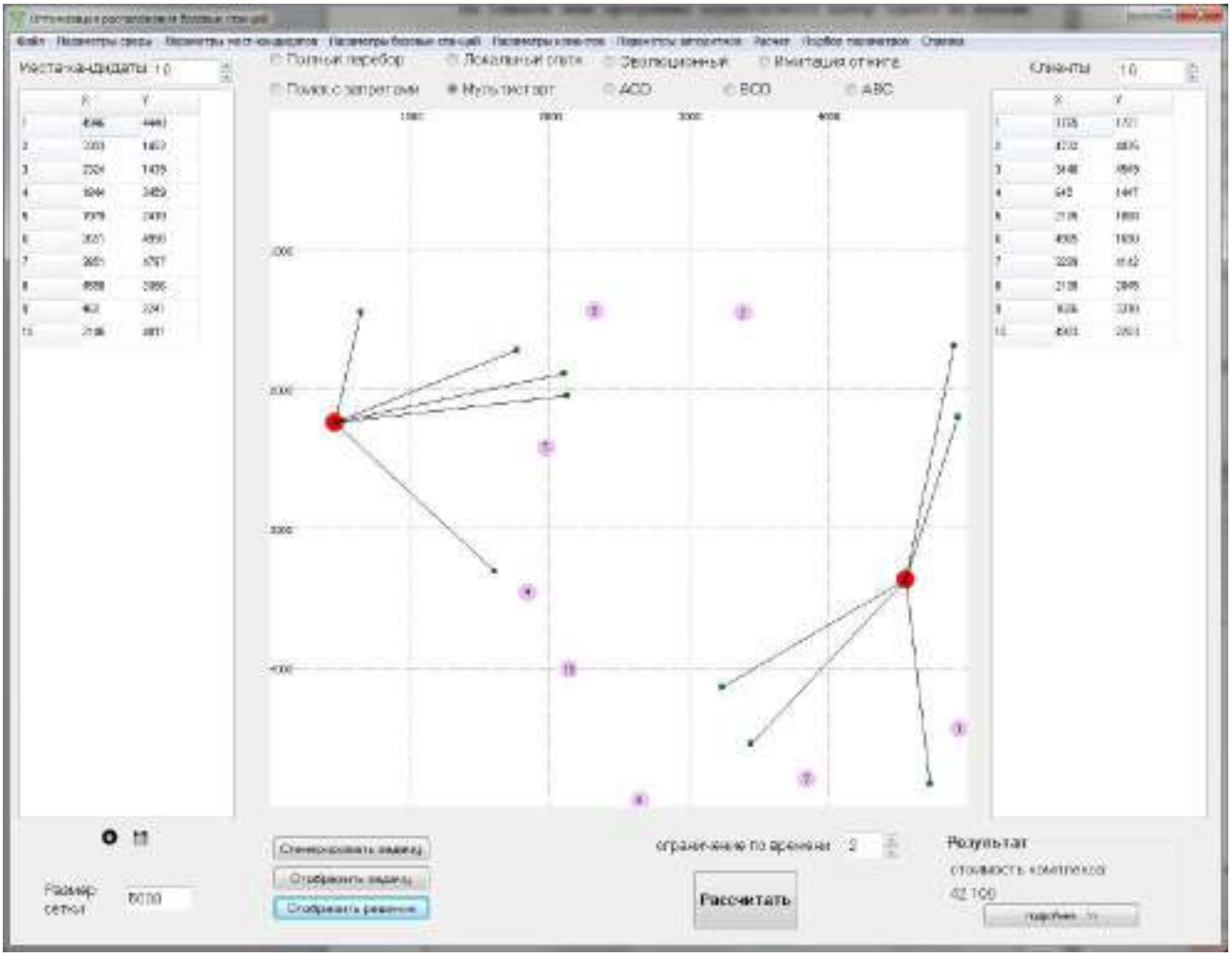

Рисунок П3.10 – Вид главного окна программы с результатами расчетов (БС соединены с клиентами)

Для каждой БС в соответствующей строке по нажатию на ячейку с символами "..." откроется форма со списком подключенных клиентов (рисунок П3.12). Таким образом, в качестве результата работы программного комплекса, помогающего лицу, принимающему решение, оптимизировать размещение БС при проектировании беспроводных сетей передачи данных, выступает выдаваемая пользователю информация о необходимых местах размещения базовых станций, типах БС, местах с которых можно удалить БС, а также рекомендации по модернизации БС.

Если пользователь хочет запустить не один расчет, а сразу несколько, то он может сделать это при помощи формы запуска серии вычислительных экспериментов (рисунок 4.3).

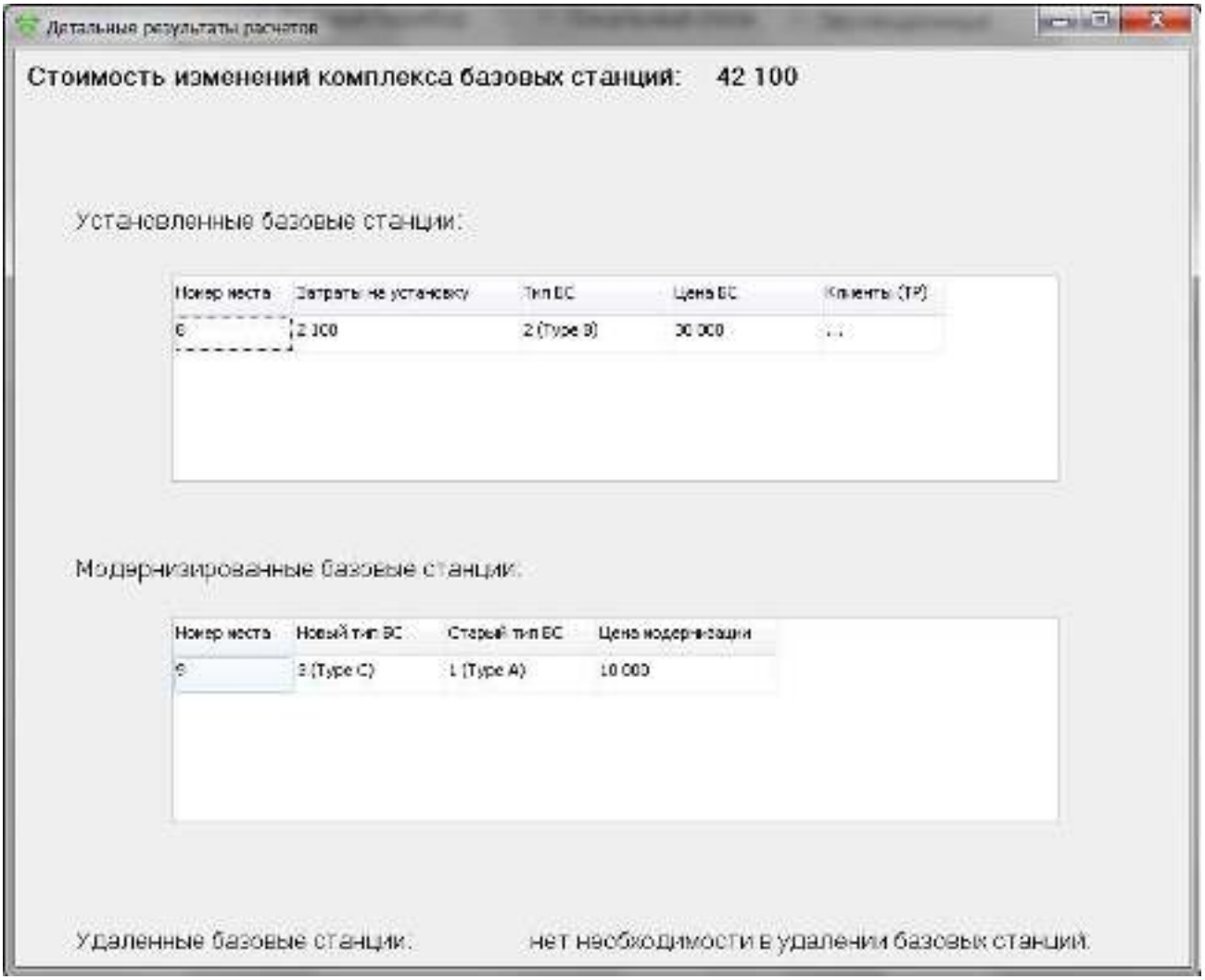

Рисунок П3.11 – Вид формы «Детальные результаты расчетов»

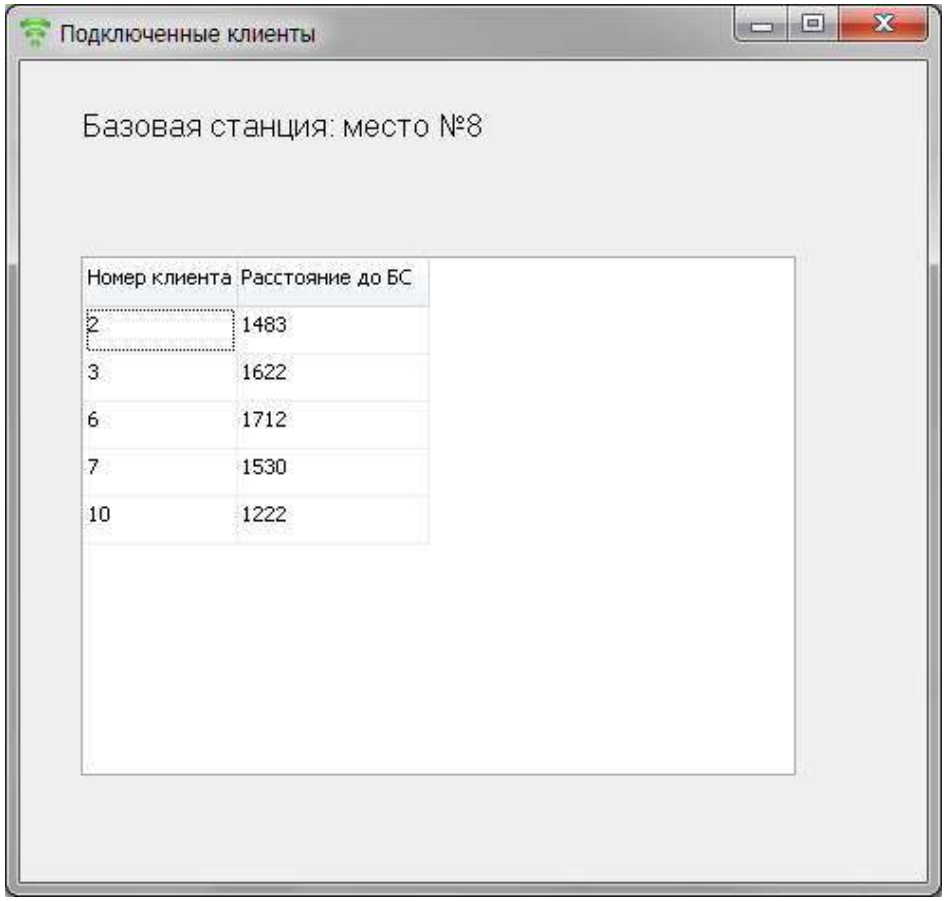

Рисунок П3.12 – Вид формы «Подключенные клиенты»

Отдельно стоит сказать о функционале программы, позволяющем осуществлять настройку параметров метаэвристических алгоритмов решения задачи размещения элементов развивающихся информационных систем. Данная форма (рисунок П3.13) вызывается нажатием пункта меню «Настройка параметров алгоритмов». Пользователь имеет возможность выбрать один из семи алгоритмов решения ЗРЭРИС. Затем пользователю необходимо ввести множества допустимых значений параметров выбранного алгоритма, а также задать время, отводимое на настройку параметров. Настройка значений параметров (первая его фаза) начнется по нажатию кнопки «Начать настройку». По окончанию расчетов программа выдаст пользователю найденные параметров выбранного алгоритма (рисунок П3.14). При желании пользователь имеет возможность запустить вторую фазу настройки параметров, которая заключается в более детальных поисках в окрестности значений параметров, найденных на первой фазе расчетов.

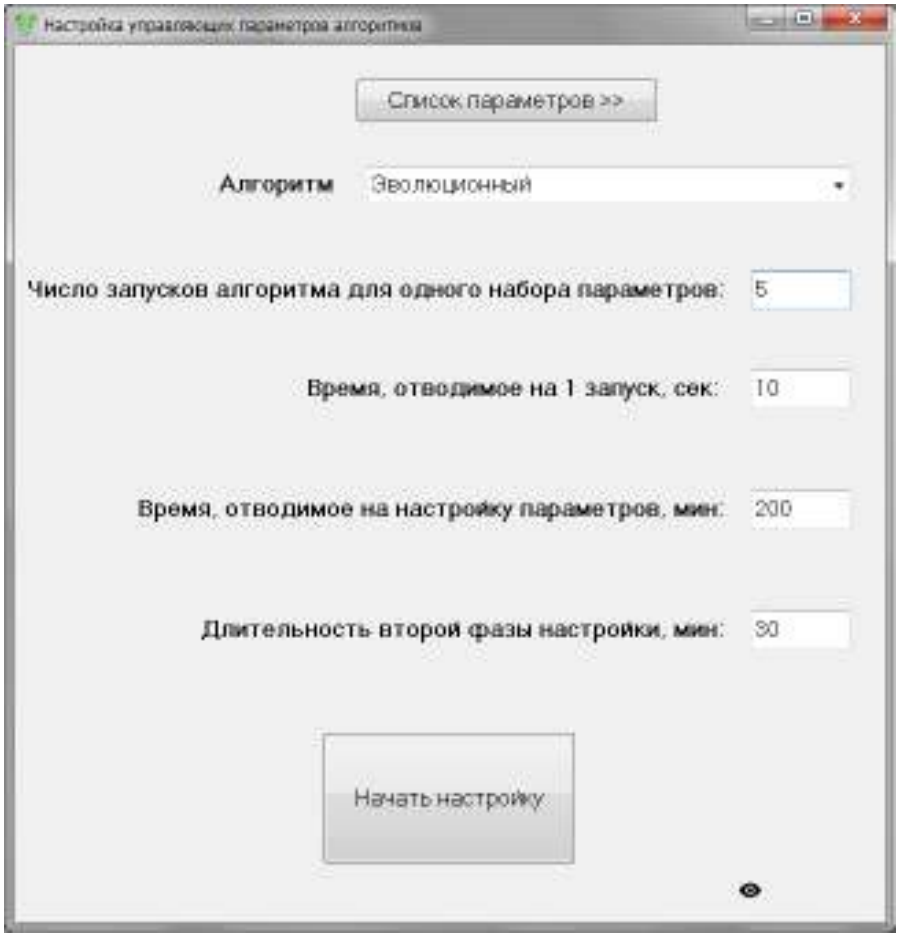

Рисунок П3.13 – Вид формы «Настройка управляющих параметров алгоритмов»

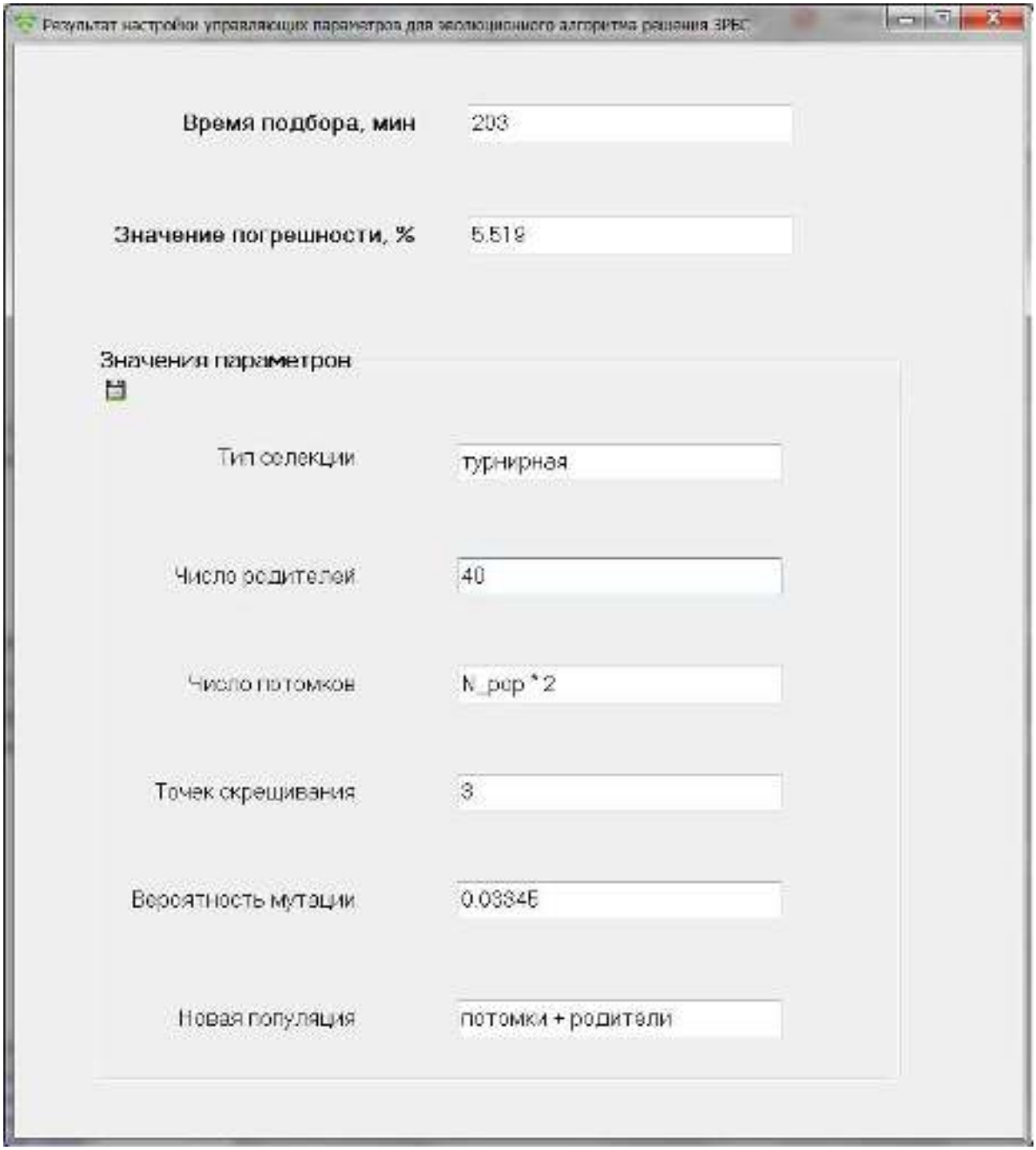

Рисунок П3.14 – Вид формы «Результат настройки управляющих параметров для эволюционного алгоритма решения ЗРЭРИС»

ПРИЛОЖЕНИЕ 4. АКТЫ ВНЕДРЕНИЯ РЕЗУЛЬТАТОВ КАНДИДАТСКОЙ ДИССЕРТАЦИИ

УТВЕРЖДАЮ
и.о. ректора
ФГБОУ ВО «Липецкий государственный
педагогический университет
имени П.П. Семенова-Тян-Шанского»
к.в.н., доцент Федина Н.В.

«27» апреля 2017 года

Акт внедрения
результатов кандидатской диссертации
Скакова Евгения Сергеевича
в учебный процесс ФГБОУ ВО
«Липецкий государственный педагогический университет имени
П.П. Семенова-Тян-Шанского»

Настоящим актом удостоверяется, что результаты исследований, полученные в диссертации Скакова Е.С., внедрены в учебный процесс по специальности 01.03.02 – «Прикладная математика и информатика».

Разработанные в диссертационной работе методы и алгоритмы интеллектуальной поддержки принятия решений по оптимизации размещения элементов развивающихся информационных систем использованы при изучении студентами дисциплин «Методы теории принятия решений» и «Методы оптимизации». Рассмотренные в диссертации алгоритмы, методы, подходы и концепции используются студентами при выполнении курсовых и дипломных работ

Директор института естественных,
математических и технических наук
к.ф.-м.н., доцент М.Ю. Смирнов

Рисунок П4.1 – Акт о внедрении. ФГБОУ ВО «Липецкий государственный педагогический университет имени П.П. Семенова-Тян-Шанского»

Открытое Акционерное Общество «Мобильные ТелеСистемы»
Филиал в Липецкой области
398046, г. Липецк, проспект 60 лет СССР, д. 18
тел. (4742) 39-69-61
факс: (4742) 39-30-26

«27» мая 2015 г.

АКТ
о внедрении результатов диссертационной работы
Скакова Евгения Сергеевича

Настоящим подтверждается, что результаты диссертационной работы Скакова Евгения Сергеевича, представленной на соискание ученой степени кандидата технических наук и посвященной проблеме размещения базовых станций в беспроводных сетях, обладают актуальностью, представляют практический интерес и были использованы ОАО "Мобильные ТелеСистемы" при проведении работ по планированию и оптимизации радиосетей стандарта 3G/4G.

При этом в частности использованы:

– разработанные автором метаэвристические алгоритмы оптимального размещения базовых станций (эволюционный, имитации отжига, поиска с запретами, мульти-старта);

– программное обеспечение, созданное для автоматизации процесса построения сетей беспроводного доступа.

Использование результатов диссертационной работы Скакова Е.С. при проведении указанных работ способствовало их успешному и качественному выполнению в части, относящейся к выбору мест для размещения базовых станций, что позволило уменьшить капитальные затраты оператора на создание сети и обеспечило требуемые параметры качества обслуживания для подключенных клиентов.

Созданное программное обеспечение является универсальным продуктом, который может быть применен при планировании любой беспроводной сети передачи данных, использующей базовые станции.

Технический директор Шеварухин Д.А.

Рисунок П4.2 – Акт о внедрении. ОАО «Мобильные ТелеСистемы», филиал в Липецкой области

АО «ЭР-Телеком Холдинг»
Филиал в г. Липецк
398016, г. Липецк, ул. Космонавтов, д. 20а
тел. +7 (4742) 555-005
факс: +7 (4742) 555-090

«2» марта 2016 г.

АКТ
о внедрении результатов диссертационной работы
Скакова Евгения Сергеевича

Настоящим актом подтверждается, что результаты диссертационной работы Скакова Евгения Сергеевича, представленной на соискание ученой степени кандидата технических наук и посвященной проблеме планирования беспроводных сетей передачи данных, внедрены в филиале в г. Липецк АО «ЭР-Телеком Холдинг».

Разработанные Скаковым Е.С. метаэвристические алгоритмы оптимального размещения базовых станций (эволюционный, имитации отжига, поиска с запретами, мульти-старта, муравьиный, пчелиный) успешно прошли апробацию и были внедрены при проведении работ по планированию и оптимизации беспроводных сетей стандарта 3G/4G.

Вышеперечисленные алгоритмы были реализованы Скаковым Е.С. в виде программного комплекса, использованного в филиале в г. Липецк АО «ЭР-Телеком Холдинг» для автоматизации процесса построения сетей беспроводного доступа.

Использование результатов диссертационной работы Скакова Е.С. при проведении работ по планированию и оптимизации беспроводных сетей позволило уменьшить капитальные затраты оператора на создание сети и обеспечило требуемые параметры качества обслуживания для подключенных клиентов.

Директор филиала в г. Липецк
АО «ЭР-Телеком Холдинг» Целовальников Е.А.

Рисунок П4.3 – Акт о внедрении. АО «ЭР-Телеком Холдинг», филиал в г. Липецк

ПРИЛОЖЕНИЕ 5. СВИДЕТЕЛЬСТВА О ГОСУДАРСТВЕННОЙ РЕГИСТРАЦИИ ПРОГРАММЫ ДЛЯ ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015611925

«Эволюционный алгоритм решения задачи размещения базовых станций»

Правообладатель: *Скаков Евгений Сергеевич (RU)*

Автор: *Скаков Евгений Сергеевич (RU)*

Заявка № 2014662999
Дата поступления 12 декабря 2014 г.
Дата государственной регистрации
в Реестре программ для ЭВМ *09 февраля 2015 г.*

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирин

Рисунок П5.1 – Свидетельство о государственной регистрации программы «Эволюционный алгоритм решения задачи размещения базовых станций»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015618283

«Программа для решения задачи размещения базовых станций методами поиска с запретами и мульти-старта»

Правообладатель: ***Скаков Евгений Сергеевич (RU)***

Автор: ***Скаков Евгений Сергеевич (RU)***

Заявка № **2015615260**
Дата поступления **09 июня 2015 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***05 августа 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Рисунок П5.2 – Свидетельство о государственной регистрации программы «Программа для решения задачи размещения базовых станций методами поиска с запретами и мульти-старта»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2016613530

«Программа для решения задачи планирования беспроводных сетей передачи данных при помощи метаэвристических методов оптимизации»

Правообладатель: *Скаков Евгений Сергеевич (RU)*

Автор: *Скаков Евгений Сергеевич (RU)*

Заявка № **2016610856**
Дата поступления **04 февраля 2016 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***29 марта 2016 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Рисунок П5.3 – Свидетельство о государственной регистрации программы «Программа для решения задачи планирования беспроводных сетей передачи данных при помощи метаэвристических методов оптимизации»