Федеральное государственное автономное образовательное учреждение высшего образования «Крымский федеральный университет имени В. И. Вернадского» (ФГАОУ ВО «КФУ им. В. И. Вернадского»)

На правах рукописи

Германчук Мария Сергеевна

Знаниеориентированные модели многоагентной маршрутизации

Специальность 05.13.18 —
«Математическое моделирование, численные методы и комплексы программ»

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
кандидат физико-математических наук, доцент
Козлова Маргарита Геннадьевна

Симферополь — 2022

Оглавление

Стр.

Стр.

Введение

Актуальность темы. Задача коммивояжера или Traveling Salesman Problem (*TSP*) имеет значимую историю теоретических исследований, алгоритмических разработок, обобщений и широких приложений. *TSP* является NP-трудной в сильном смысле, поэтому на ней проверяются новые идеи решения NP-трудных задач комбинаторной оптимизации (*ЗКО*). С другой стороны, *TSP* входит в модели маршрутизации вместе с другими задачами дискретной оптимизации. С *TSP* связаны классические красивые и сложные результаты по теории графов, решению экстремальных задач на них. Современные направления исследований относятся к многоагентным задачам маршрутизации (multiple *TSP* или $mTSP$) на сложных сетях (т. е. на графах, элементам которых приписаны некоторые величины: вес, длина, интенсивность) большой размерности и сложной структуры. Такие задачи относятся к классу задач дискретной оптимизации (*ЗДО*).

Объект исследования — задачи условной дискретной оптимизации.

Предмет исследования — многоагентные задачи маршрутизации типа TSP, $mTSP$.

Степень разработанности темы исследования. *ЗДО* возникают при планировании производства; оптимизации коммуникационной инфраструктуры; оптимизации работы исполнителей, агентов-коммивояжеров. Разработке и исследованию алгоритмов *ЗДО* и *ЗКО* посвящены труды А. А. Агеева, А. Ахо, В. Л. Береснева, Г. В. Генса, Э. Х. Гимади, Н. И. Глебова, А. Ю. Горнова, В. Г. Дейнеко, В. А. Емеличева, А. Н. Еремеева, Я. М. Ерусалимского, Ю. И. Журавлева, А. В. Кельманова, Д. Кнута, М. Я. Ковалева, А. Н. Колмогорова, А. А. Колоколова, А. А. Корбута, Ю. А. Кочетова, Н. Н. Кузюрина, А. А. Лазарева, Е. В. Левнера, В. К. Леонтьева, М. Либуры, Вл. Д. Мазурова, А. Б. Муравника, Дж. фон Неймана, В. А. Перепелицы, В. К. Попкова, С. В. Севастьянова, А. Л. Семенова, И. В. Сергиенко, И. Х. Сигала, В. А. Скороходова, А. Фиакко, М. Ю. Хачая, S. Arora, G. Cornuéjols, D. Hochbaum, B. Korte, B. K. Lovász, P. Raghavan, C. Thompson и многих др.

Среди методов решения *ЗДО*, *ЗКО* (методов отсечений, динамического программирования, ветвей и границ, декомпозиции, метода множителей Лагранжа) выделяются методы и алгоритмы локального поиска. Основополагающие результаты этого направления получены Ю. И. Журавлевым, который впервые ввел понятие и рассмотрел класс локальных алгоритмов, а также оценил их сложность. Дальнейшее развитие методов локального поиска получено А. Н. Антамошкиным, Ю. А. Кочетовым, О. Э. Семенкиной, B. Kernigahan, S. Lin, C. Papadimitriou, C. Tovey, M. Yannakakis и др.

Несмотря на множество результатов для $mTSP$ есть ряд направлений, требующих исследования.

Необходимо отметить, что массовые задачи *TSP*, $mTSP$ являются NP-полными. При этом индивидуальные экземпляры могут быть эффективно разрешимы с помощью точных, эвристических или комбинированных алгоритмов. Распознавать экземпляры задач с экспоненциальной или полиномиальной сложностью до численных расчетов практически

невозможно. Исходя из того, что задача $mTSP$ является моделью некоторой реальной прикладной задачи, можно за счет переформулировки и снятия некоторых ограничений сразу представлять входные данные в виде, приводящем к полиномиально разрешимым задачам TSP или $mTSP$ и отдельно фиксировать ограничения, сильно ухудшающие разрешимость задачи, а также отдельно обрабатывать хорошие экземпляры и заведомо плохие. Такой подход требует формирования базы полиномиально разрешимых задач, их признаков, алгоритмов распознавания или формирования исходных данных в виде, допускающем полиномиальную разрешимость. Существенной является возможность снижения размерности этапа преобразований с экспоненциальной сложностью. Например, с помощью использования иерархичности модели, кластеризации сети, декомпозиции исходной задачи, распараллеливания процесса реализации алгоритмов. Используемые знания об исходной прикладной задаче, ее модели в виде условной *ЗДО* могут быть представлены в удобной форме, например, в виде продукций. При этом необходимо сочетание системы логического вывода с прецедентной информацией, синтезом продукционной системы знаний на базе прецедентной информации. В этой части подход опирается на исследования многих авторов и, в частности, на работы В. И. Донского, М. Г. Козловой. Для задач $mTSP$ в виде псевдобулевой оптимизации такой подход реализован в работе.

Цель и задачи исследования. Целью работы является разработка прикладных моделей задач и алгоритмов многоагентной маршрутизации типа многих коммивояжеров в сложных сетях с учетом данных, фактов и знаний о структуре сети, специфике и ограничениях на прохождение маршрутов, имеющихся прецедентов.

Такая целевая установка предполагает решение следующих задач:

1. Структурировать известные и предложить новые обобщенные математические модели TSP или $mTSP$, в зависимости от возможности использования знаний о задаче, структуре сети, прецедентах.

2. Обосновать сведение обобщенной модели $mTSP$ к модели псевдобулевой оптимизации с ограничениями в виде дизъюнктивной нормальной формы (*ДНФ*) для теоретического обоснования методов и алгоритмов решения однокритериальных и многокритериальных задач маршрутизации.

3. Предложить схему по упрощению исходной задачи $mTSP$; сформировать набор алгоритмов, реализующих данную схему.

4. Провести программную реализацию набора алгоритмов решения задач TSP и $mTSP$; реализовать подход, основанный на кластеризации сети с последующим применением реализованных алгоритмов и провести квазиреальные эксперименты по анализу применяемых алгоритмов.

5. Применить предложенные модели и алгоритмы при разработке комплексов программ для решения прикладных задач.

Материалы и методы исследования. При выполнении работы применялись методы теории графов, дискретной оптимизации, математического и псевдобулевого программирования, методы кластеризации; метаэвристики и эволюционные алгоритмы, а также методология квазиреальных вычислительных экспериментов.

Положения, выносимые на защиту, и их научная новизна.

1. Выделение класса постановок модельных задач TSP, $mTSP$ и алгоритмов их решения, пригодных для синтеза комбинированных алгоритмов, в которых учитываются: прикладной характер математических моделей, знания о модели и сложной структуре сети, прецедентные знания и возможность реоптимизации.

2. Обоснование представления $mTSP$ как модели псевдобулевой оптимизации, в которой часть ограничений (или все) могут быть заданы в дизъюнктивной нормальной форме, для которых процедура поиска и логического вывода о принадлежности к искомому решению является полиномиальной.

3. Процедура снижения размерности исходной задачи $mTSP$ с помощью кластеризации сложных сетевых структур и итерационного уточнения кластеров в зависимости от решения TSP на каждом кластере и в целом. Численная реализация алгоритмов и проведение вычислительных экспериментов по кластеризации сети и построение решения $mTSP$.

4. Программные комплексы многоагентной маршрутизации для решения задач выбора наилучших туристических маршрутов по Крыму, многоагентной инфраструктурной маршрутизации в чрезвычайных ситуациях ($ЧС$), для исследования влияния политических мемов на пользователей Рунета.

Достоверность научных положений и выводов. В работе применялись математически обоснованные методы. Теоретически обосновано сведение задачи $mTSP$ к задаче псевдобулевой оптимизации с ограничением в виде $ДНФ$ ограничением. Приближенные решения подтверждены вычислительными экспериментами на основе разработанных алгоритмов.

Теоретическая и практическая значимость состоит в теоретическом обосновании построения маршрутов многих коммивояжеров в одно- и многокритериальных постановках на базе сведения к задачам псевдобулевой дискретной оптимизации с $ДНФ$ ограничениями.

Предложено алгоритмическое наполнение и программная реализация в рамках схемы снижения размерности с помощью декомпозиции, согласованной с $mTSP$. Результаты применены для разработки программных комплексов прикладных задач: многоагентной инфраструктурной маршрутизации в $ЧС$, построению наилучших туристических маршрутов и исследованию влияния политических мемов на пользователей Рунета («Memometrix»). Получены свидетельства о регистрации программ для ЭВМ «Программа многоагентной инфраструктурной маршрутизации» № 2022614174 от 17.03.2022 г., «Программа выбора наилучших туристических маршрутов по Крыму» № 2021681822 от 27.12.2021 г. и акт о внедрении результатов диссертационного исследования.

Апробация работы. Результаты исследования были представлены на следующих конференциях: Всероссийская конференция с международным участием «Математические методы распознавания образов» (г. Москва, 2019 г., 2021 г.), 13-я Международная конференция «Интеллектуализация обработки информации» (г. Москва, 2020 г.), Международная научно-техническая конференция «Радиолокация, навигация, связь» (г. Воронеж, 2020 г., 2021 г.), Международная школа-симпозиум «Анализ, моделирование, управление, развитие социально-экономических систем» (г. Судак, 2015 г., 2017–2020 гг.), Научно-практическая

конференция «Математика, информатика, компьютерные науки, моделирование, образование» и Таврическая научная конференция студентов и молодых специалистов по математике и информатике (г. Симферополь, 2017–2019 гг.), Всероссийская научно-практическая конференция (с международным участием) «Дистанционные образовательные технологии» (г. Ялта, 2018–2020 гг.), Международная открытая конференция «Современные проблемы анализа динамических систем. Теория и практика» (г. Воронеж, 2019 г.), III Международная научная конференция «Осенние математические чтения в Адыгее» (Майкоп, 2019 г.), V Международная научно-практическая конференция «Информационные системы и технологии в моделировании и управлении» (г. Ялта, 2020 г.), научный семинар кафедры информатики (г. Симферополь, 2016 г., 2022 г.)

Публикации. Основные результаты работы опубликованы в рекомендуемых ВАК РФ рецензируемых научных изданиях [18; 22; 28; 33; 34; 36; 210]. Проиндексированы в Scopus [185; 186; 211].

Личный вклад автора. Автором диссертации совместно с научным руководителем проводилась постановка задач, обсуждались полученные основные результаты и формулировки выводов. Лично автором доказана сводимость задачи $mTSP$ к задаче псевдобулевой оптимизации с *ДНФ* ограничением. Автором обоснован подход к решению задач $mTSP$ на базе логической системы продукций. В силу сложности $mTSP$ на инфраструктурных сетях предложена схема снижения размерности задачи. Разработанные и апробированные в работе алгоритмы применены при разработке программных комплексов «Программа многоагентной инфраструктурной маршрутизации», «Программа выбора наилучших туристических маршрутов по Крыму».

Область исследования соответствует следующим пунктам паспорта специальности 5.13.18 — «Математическое моделирование, численные методы и комплексы программ» (физико-математические науки):

1. Разработка новых математических методов моделирования объектов и явлений.

3. Разработка, обоснование и тестирование эффективных вычислительных методов с применением современных компьютерных технологий.

4. Реализация эффективных численных методов и алгоритмов в виде комплексов проблемно-ориентированных программ для проведения вычислительного эксперимента.

Объем и структура работы. Диссертация состоит из введения, 4 глав, заключения.

Полный объем диссертации составляет 150 страниц, включая 49 рисунков и 2 таблицы. Список литературы содержит 255 наименований.

Глава 1. Модели и задачи многоагентной маршрутизации

1.1 Задачи коммивояжера и их обобщения

Классическая задача коммивояжера (TSP) и ее методы решения достаточно востребованы. Менее известны ее обобщения, которые возникают в различных приложениях. Рассмотрим знаниеориентированные модели TSP, в которых представляется важным учет информации и знаний, приписываемых вершинам, дугам, числу коммивояжеров (m коммивояжеров — $mTSP$ (multiple Traveling Salesman Problem)) и их взаимодействиям. Ограничимся обобщениями TSP, $mTSP$, которые имеют отношение к приложениям, рассматриваемым в работе.

Сформулируем рабочее определение.

Определение 1.1. *Знаниеориентированными называются такие задачи TSP или $mTSP$, для решения которых используются представления знаний о структуре сети, априорные, прецедентные, аппроксимационные*; *знания о решении близких задач, а также знания, приводящие к выделению полиномиально разрешимым классам задач, которые могут быть представлены в любой удобной форме, например, в виде логической системы продукций.*

TSP нахождения оптимального маршрута обхода вершин графа с заданной матрицей попарных стоимостей перемещения между вершинами является классической NP-трудной задачей с широкими приложениями [9; 69; 89; 110; 132]. Рассматриваемый в работе класс задач является развитием TSP. Обзор теоретических результатов, точных и приближенных алгоритмов содержится в работах [94—96]. Приведем сетевую формулировку TSP. Пусть $G = (V,U)$ — ориентированный (неориентированный) граф, где V ($|V| = n$) — множество вершин, U — множество дуг (ребер). Для ориентированных графов приняты термины: дуга, путь, контур. Для неориентированных графов — ребро, цепь, цикл. Предполагается, что графы без петель и кратных дуг (ребер). Пусть C — матрица $n \times n$ действительных чисел, $c_{ij} \geqslant 0$, $(i,j) \in U$ длин дуг (весовые коэффициенты). В TSP требуется найти замкнутый маршрут (т. е. начинающийся и заканчивающийся в одном городе) коммивояжера, проходящий через все города (вершины графа) по одному разу и имеющий минимальную длину.

Маршрут, соединяющий вершины i и j, — конечная непрерывная последовательность ребер и инцидентных им вершин с концами в i и j. Он называется замкнутым, если $i = j$. Маршрут называется цепью, если его ребра разные, и простой цепью, если все вершины разные. Цикл — это замкнутая цепь, а простой цикл — это замкнутая простая цепь. Общая TSP состоит в поиске цикла минимальной стоимости, проходящего по всем вершинам графа G. Простой цикл, проходящий по всем вершинам G один раз, называется гамильтоновым, а соответствующая задача — гамильтоновой задачей коммивояжера ($GTSP$). Для случая, когда граф является метрическим и полным, существуют быстродействующие приближенные

алгоритмы решения [17; 101] *GTSP*. Рассмотрим некоторые обобщения (модель уточняется на основе дополнительной информации, знаний).

Логистические структуры, индуцированные практическими задачами, создают разнообразие сетевых структур и задач. В случае *гамильтоновой задачи сельского почтальона* дополнительно задается множество ребер $S \subset U$, через которые обязательно должен пройти гамильтонов цикл. Аналогичная задача возникает при моделировании процесса построения туристических маршрутов и маршрутов агентов в чрезвычайных ситуациях по поставке ресурсов (см. главу 4).

В *задаче инкассатора*, обобщающей *TSP*, необходимо найти маршрут при заданном (или оптимальном) сроке и при минимизации затрат на перевозки, т. е. задается информация (веса) о ребрах и вершинах исходного графа, выделяется множество $S \subset U$ ребер обязательных для включения в маршрут. Естественными являются критерии: минимум числа бригад (коммивояжеров), суммарных отклонений от установленных сроков доставок, простоя, времени нахождения на маршруте, пройденного расстояния, риска перевозки груза (например, как произведения времени на стоимость груза); максимум вероятности успешной доставки в заданный интервал времени прибытия коммивояжера для выделенной вершины графа (пункте обслуживания).

Важной является задача $mTSP$ выбора m смешанных автомобильных и пеших маршрутов групп туристов. В качестве исходной информации будут достопримечательности, которые выбраны для посещения; возможные автомобильные и пешеходные маршруты, позволяющие посетить достопримечательности; точки начала пешеходных маршрутов, к которым нужно доставить группы туристов, и точки, где нужно забрать группы в конце маршрута. Требуется составить замкнутые маршруты автотранспорта (агентов) для перевозки каждой из групп туристов, посещающих выделенные достопримечательности (объекты-вершины) на пешеходных участках маршрутов и на автомобильных. Отметим, что данная задача сводится к задаче псевдобулевого программирования с учетом окон посещения достопримечательностей и ограничений, связанных с одновременным посещением группами одних и тех же объектов. При этом минимизируется длина пути автотранспорта или времени в пути. Близкая задача рассмотрена в 4.2.

Также актуальна *задача облета*. Например, для выбора маршрутов многих коммивояжеров в сложной инфраструктурной сети можно использовать беспилотный летательный аппарат (*БПЛА*), так называемый дрон. Этот вариант $mTSP$ порождает обобщение задачи многих коммивояжеров $mTSPD$. Данная задача связана с упрощением исходной сети, когда ей ставится в соответствие плоская сеть (координаты вершин задаются на плоскости облета) с метрическим и полным графом (с весами, удовлетворяющими неравенству треугольника), что, в свою очередь, приводит к полиномиальным алгоритмам $mTSP$ (см. раздел 4.1).

В достаточно общем случае необходимо решать задачи для нескольких коммивояжеров с несколькими критериями, дополнительными условиями (информация, знания, прецеденты). Например, в *задаче о поиске максимально простой цепи* между двумя вершинами (т. е. с максимальным количеством вершин) каждой вершине приписывается некоторый предикат, задающий информацию о вершине (включение вершины в цепь или исключение и т. п.).

В [133] такие условия названы срединными и приводится рекурсивный алгоритм построения цепи с проверкой условий (см. также работу Я. М. Ерусалимского и В. А. Скороходова [41]).

Для нескольких коммивояжеров возникают ситуации общей цели (согласованная работа), и тогда задача может быть сведена к случаю одного коммивояжера или конкуренции (захвата сети с учетом ресурсов), что приводит к играм на графах (соответствующий результат представлялся в работах автора [20; 67]).

Знаниеориентированная модификация *TSP* зависит от исходной информации. Например, для множества n вершин V предписывается порядок посещения некоторых вершин, т. е. задаются условия предшествования. Требуется найти решение *TSP*, не нарушая условий предшествования. В приложениях это *задача о кольцевом маршруте* (Dial-A-Ride Problem).

В работе [112] отражены направления, связанные с ограничениями, внутренними потерями и с условиями предшествования (*задача курьера*). Другие обобщения связаны с обходом конечной системы множеств и дополнительными затратами, связанными с выполнением работ. Для решения применяется оптимальный алгоритм на основе экономичной версии динамического программирования, в рамках которого и учитывается дополнительная информация. Для такого класса задач также применим метод ветвей и границ.

Возможные приложения связаны с транспортными перевозками, логистическими задачами, для которых характерны ограничения в виде условий предшествования. Транспортное средство по прибытию в пункт может принимать груз или корреспонденцию для доставки в другой пункт. Образуется пара: отправитель — получатель. Информация о таких парах формирует ограничения необходимые для исполнения, что усложняет выбор маршрута коммивояжера.

В *TSP* с множеством выделенных вершин V графа G выделяется множество $V_1 \subset V$ $(|V_1| = k; |V| = n)$. Требуется построить маршрут коммивояжера, обязательно проходящий через множество выделенных вершин V_1 точно один раз, а через остальные $n - k$ вершин — не более одного раза [26; 73].

В ряде случаев решение задачи для нескольких коммивояжеров предваряет задача кластеризации исходного графа на два или несколько подграфов.

Кластеризация необходима для декомпозиции задачи при ее большой размерности.

В *кластерной TSP* множество вершин V разбито на кластеры (компоненты) V_i, $i = \overline{1,k}$, $V_1 \cup V_2 \cup \ldots \cup V_k \subseteq V$. Возможны варианты:

1. Коммивояжер посещает все вершины кластера V_i подряд и только после этого переходит к кластеру V_j.
2. На маршрут в кластере V_i и между ними могут быть заданы условия предшествования и другие условия.
3. Требуется посетить в каждом кластере только часть вершин (или одну, минимальный маршрут между V_i, $i = \overline{1,k}$).
4. Находится представительная вершина кластера V_i такая, что расстояние до непосещаемых вершин кластера не превышает заданной величины.
5. Требуется определить кластеры, удовлетворяющие условию 4.

Обобщенная TSP состоит в построении полного кратчайшего контура, где условие посещения вершины только один раз не накладывается.

В *TSP с несколькими весовыми матрицами* кроме матрицы C задается еще одна или несколько других матриц весов дуг (ребер) [94]:

1. Матрицы задают ограничения сверху и (или) снизу на вес маршрута коммивояжера, и требуется найти решения *TSP* с матрицей C, удовлетворяющее ограничениям на веса/маршрут (кроме длины маршрута, могут быть ограничения на время прохождения маршрута, общие затраты и т. п.).
2. Матрицы могут задавать дополнительные ограничения на части маршрута коммивояжера.
3. Матрицы могут задавать критерии и ограничения.

В *динамических TSP* [1] задается n матриц C^t с элементами c_{ij}^t. Пусть в маршрут коммивояжера уже включено k вершин (городов), тогда расстояние до $k+1$ вершины необходимо искать по матрице C^k (c_{ij}^t). Требуется найти решение *TSP* с такими дополнительными условиями. Матрица C^t может задаваться по другим правилам. Практические задачи маршрутизации существенно расширяют список динамических *TSP*.

Реальные ситуации содержат близкие постановки задач, для которых важны как точные, так и приближенные решения, которые, как правило, базируются на комбинациях локальных, эвристических алгоритмов. В экстремальных задачах анализа и синтеза на графах необходимо учитывать знания, информацию, факты и прецеденты. Разнообразие задач диктуется классами графов, моделирующих транспортные сети; структурой графов, их размерностью, возможностью декомпозиции; характером целевых функций и полнотой информации о коэффициентах критериев; возможностью представления знаний об ограничениях на сети в виде дизъюнктивных нормальных форм (*ДНФ*). Использование дополнительной информации (знаний) по обязательным ограничениям усложняют задачу.

Выделение блоков в задачах $mTSP$ возможно интерпретировать как процесс кластеризации, который определяется спецификой задачи. В том случае, когда явно присутствуют кластеры (иерархическая инфраструктура города, региона и т. д.), можно использовать известные алгоритмы кластеризации. Декомпозиция задачи с помощью процедуры максимального разреза выделяет две компоненты, на которых решается *TSP*. Тем самым определяется необходимое число коммивояжеров. Если размерности полученных компонент велики, то процедура повторяется. Для приближенного решения используются эвристики, как в задаче о максимальном разрезе, так и для решения *TSP* на кластерах. Например, приближенное решение задачи предполагает предварительное извлечение информации:

1) по статистике распределения расстояний (ранжирования);

2) по грубому распределению кластеров, качественной информации о компонентах, экспертной и прецедентной информации.

Большую роль в прикладной теории сетей играет предобработка доступных данных о структуре сети. В работе [131] подчеркивается необходимость использования особенностей взвешенных графов для более быстрого определения их характеристик: центра, радиуса, диаметра. В свою очередь эти характеристики можно вычислять по матрице кратчайших

расстояний. Экстремальные задачи на графах большой размерности исследуются в [2]. Алгоритмы поиска путей на графах большого размера тесно связаны с задачей разбиения графа на кластеры [77; 104; 134], а также с рациональным способом хранения больших графов [104]. Задача о кратчайшем пути для графа большой размерности в [11] связывается с адаптивным размещением ориентиров.

Специфика алгоритмического получения элементов сети и построения замкнутых маршрутов, соответствующим циклам динамических систем, исследовалась автором в работах [37; 38; 90]. Отметим необходимость использования большого числа ребер сети (миллионы) и эффективность алгоритмов. Изложение данного класса задач здесь не приводится ввиду необходимости введения большого объема предварительной информации.

1.1.1 Подклассы полиномиально разрешимых задач коммивояжера

В приложениях при формализации задач TSP или $mTSP$ на сетях большой размерности и сложной структуры часто пренебрегают спецификой, особенностями и некоторыми ограничениями. Может оказаться, что более точный или избирательный учет знаний о задаче и сети, позволит выделить индивидуальный экземпляр задачи, разрешимой за полиномиальное время или класс таких задач. Существует путь формализации задачи и организации структуры исходных данных, при котором полученный экземпляр будет полиномиально разрешимым. Метод, в котором исходные данные представлены определенным образом в работе [49], называется методом моделирования исходных данных. Метод, который основан на распознавании структуры входных данных и заданном упорядочении комбинаторных комбинаций [128], называют методом структурно-алфавитного поиска оптимального решения задач комбинаторной оптимизации.

Пусть C — весовая матрица расстояний с элементами c_{ij}, $i,j = \overline{1,n}$ из класса R^n симметричных $n \times n$ матриц над полем вещественных чисел $\mathbb{R}$. Постановка TSP, как комбинаторной задачи оптимизации, состоит в нахождении перестановки π на множестве индексов $\{1,2,\ldots,n\}$, которая минимизирует функцию расстояния (весовая функция)

$$f = \sum_{i=1}^{n} c_{i\pi(i)}. \tag{1.1}$$

То есть коммивояжер должен объехать все города (вершины) в произвольной последовательности и вернутся в исходный город кратчайшим путем $\gamma = (1,\pi(1),\pi(2),\ldots,\pi(n) = 1)$.

Определение 1.2. *Через PS обозначим подкласс NP-полных задач коммивояжера, которые полиномиально разрешимы при дополнительных ограничениях (условиях, знаниях) S.*

Приведем некоторые примеры TSP из подкласса PS. Для плоской TSP полиномиально разрешимой является задача, элементы матрицы C которой удовлетворяют неравенству треугольника. Другие подобные подклассы определяются через четыре различных элемента матрицы C. Пусть $1 \leqslant i < j < k < l \leqslant n$, $C \in R^n$ произвольная матрица расстояний.

Симметричная матрица $C \in R^n$ называется матрицей Кальмансона ($\mathcal{K}$) [202], если выполняется условия

$$\theta_1 = c_{ij} + c_{kl} \leqslant c_{ik} + c_{jl} \equiv \theta_2, \quad \theta_3 \equiv c_{ij} = c_{jk} \leqslant c_{ik} + c_{jl} \equiv \theta_2. \tag{1.2}$$

или $\theta_2 = max\{\theta_1, \theta_2, \theta_3.\}$ Если $C \subset \mathcal{K}$, то оптимальным маршрутом TSP будет $\gamma = (1,2,\ldots,n-1,n)$.

Матрицы Супника ($\mathcal{S}$) [245] удовлетворяют условиям

$$\theta_1 \leqslant \theta_2, \quad \theta_3 \geqslant \theta_2 \tag{1.3}$$

или $\theta_1 = min\{\theta_1, \theta_2, \theta_3.\}$ Для матриц Супника ($\mathcal{S}$) оптимальным маршрутом будет $\gamma = (1,3,5,7,\ldots,8,6,4,2)$. Более мощным является класс матриц Демиденко [47; 48] $\mathcal{K} \cup \mathcal{S} \in \mathcal{D}$.

Пирамидальным называется такой маршрут TSP $\gamma = (1,i_1,i_2,\ldots,i_{k-1},n,i_{k+1},i_{k+2},\ldots,i_{n-2})$, в котором $i_1 < i_2 < \ldots < i_{k+1}$, и $i_{k+1} > i_{k+2} > \ldots > i_{n-2}$. В. М. Демиденко доказал, что для матрицы $C \subset \mathcal{D}$ существуют оптимальный маршрут, который является пирамидальным. Число пирамидальных маршрутов $\geqslant 2^{n-2}$. Лучший из них можно найти полиномиальным алгоритмом порядка $O(n^2)$ [126].

Произвольная матрица $C \in \mathbb{R}^n$ называется матрицей Монжа, если $\theta_1 \equiv c_{ij}+c_{kl} \leqslant c_{il}+c_{kj}$ для всех $1 \leqslant i < k \leqslant n$, $1 \leqslant j < l \leqslant n$. Матрица C не обязательно симметрическая. Задача коммивояжера с переупорядоченной матрицей расстояний Монжа называется задачей коммивояжера Монжа [187].

Возникает задача распознавания, которая состоит в том, чтобы найти перестановку δ элементов матрицы C такую, что новая матрица $\delta(i)\delta(j)$ принадлежала бы к разрешимым TSP (Демиденко, Кальмансона, Супника, Монжа и др.). В [49] утверждается, что до сих пор не доказано существование такой перестановки для конкретной матрицы расстояний C.

Заметим, что подкласс разрешимых задач с матрицей Монжа найден в результате разной перестановки для строк и столбцов: $\delta(i)\tau(j)$, δ — перестановка для строк, а τ — для столбцов.

Дальнейшее описание полиномиально разрешимых TSP связано с функцией натурального аргумента $\varphi(t)$ [49; 126; 128], с помощью которой представляются элементы матрицы C. Для симметрической матрицы C достаточно использовать наддиоганальные элементы, которые представляются построчно в виде одномерного массива c_{11}, c_{13}, …, c_{1n}, c_{23}, c_{24}, …, c_{2n}, c_{34}, …, $c_{n-1,n}$. Такой последовательности ставится в соответствии функция натурального аргумента $\varphi(t) = \{\varphi(1),\varphi(2),\ldots,\varphi(m)\}$, $m = \dfrac{1}{2}n(n-1)$. Для $j > i$ имеет место соответствие $\varphi(t) = c_{ij}$, $t = n(i-1) - \dfrac{i(i+1)}{2} + j$. Справедливы утверждения [49]:

1. Если $\varphi(t)$ — линейная функция с отрицательным угловым коэффициентом, то соответствующая TSP принадлежит к подклассу разрешимых задач.
2. Если $\varphi(t)$ — выпуклая и невозрастающая функция, то соответствующая TSP принадлежит подклассу разрешимых задач.

Матрица расстояний, представленная линейной функцией $\varphi(t)$, является одновременно матрицей Кальмансона и Супника. Впервые линейную функцию в качестве расстояний

предложила Н. К. Тимофеева в своей диссертации. В [126] взята функция расстояний, удовлетворяющая условиям: $|j-i| < |l-k| \Rightarrow c_{ij} < c_{kl}$, $c_{ij} = c_{ji}$. В этом случае оптимальный маршрут является пирамидальным циклом.

В [128] для подклассов разрешимых задач TSP доказано, что методом структурно-алфавитного поиска построением не более чем $\frac{n^2}{2}$ перестановок находится упорядоченная последовательность локальных экстремумов $f = (f(\delta^1), f(\delta^2), \ldots, f(\delta^{k*}))$ таких, что $f(\delta^{i*}) = \underset{\delta^k}{globextr} f(\delta^k)$, где k^*, $i^* \in \left(1, \ldots, \frac{n^2}{2},\right)$ $f \in \{1, \ldots, n!\}$, δ^k — перестановка, которая в TSP является аргументом целевой функции.

Если входные данные заданы функциями натурального аргумента, которые изменяются как монотонные, унимодальные (вогнутые или выпуклые), периодические или наименьшие элементы матрицы C одинаковые и образуют гамильтонов цикл, то на основании этого утверждения приводится алгоритм [128, с. 34—35] («вычислительная схема») решения TSP. Соответствующие правила отсечения неэффективных вариантов основываются на закономерности изменения значений целевой функции f от упорядочивания комбинаторных комбинаций и от структуры входных данных.

Выбор алгоритма структурно-алфавитного поиска, наряду с алгоритмами метода ветвей и границ, динамического программирования, последовательного анализа вариантов для прикладных задач многих коммивояжеров обусловлен его быстродействием и сравнимой точностью.

В нашем случае для задачи $mTSP$ эффективность применения алгоритма связывается с выбором кластера (фрагмента сети), на котором решают задачу один из агентов-коммивояжеров, а также с предварительным исследованием структуры входных данных TSP на этом кластере. Прецедентные знания позволяют накапливать базу шаблонов, т. е. выделять случаи структур данных, на которых целевые функции изменяются, и к таким TSP применимы одинаковые схемы решения. Поиск шаблонов можно осуществить с помощью технологий машинного обучения (искусственных нейронных сетей) [100]. На входе: комбинации элементов матрицы C. На выходе: оценки принадлежности C к одному из классов комбинаторных функций $\varphi(t)$. Такой подход может привести к более широкому подклассу полиномиально разрешимых TSP. Для $mTSP$ на каждом кластере может быть свой алгоритм решения.

Использование функции $\varphi(t)$ в представлении элементов матрицы C в виде строки может применяться в представлении $mTSP$ как модели псевдобудевой оптимизации (см. также гл. 2).

Реальные инфраструктурные сети являются сложными сетями большой размерности. Как было отмечено выше, для маршрутизации в сложных сетях возникает ряд уникальных задач: о нахождении метрических характеристик сложных (больших) сетей; поиск минимального (максимального) среднего пути в сети; коэффициентов кластеризации; изучения информационных потоков в сети; выявление критичных мест в сети; определения кластеров; выявление блоков, компонент, мостов, точек сочленения (перемычек).

1.2 Модели псевдобулевой дискретной оптимизации многоагентных задач коммивояжера

1.2.1 Задача о назначениях в модели псевдобулевой оптимизации

Рассмотрим задачу о назначениях в модели псевдобулевой оптимизации (целочисленного линейного программирования):

$$\begin{gathered}\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \sum_{i=1}^{n} x_{ij} = 1, \quad \sum_{j=1}^{n} x_{ij} = 1, \quad x_{ij} \in \{0,1\}, \quad 1 \leqslant i,j \leqslant n,\end{gathered} \tag{1.4}$$

которая отличается от TSP одной группой ограничений для исключения подциклов, представленных, например, в виде

$$\sum_{i\in S}\sum_{j\notin S} x_{ij} \geqslant 1, \quad S \neq \varnothing, \quad S \subset \{1,\ldots,n\} \tag{1.5}$$

Если заменить условие целочисленности (булевости) переменных $x_{ij} \in \{0,1\}$, $1 \leqslant i,j \leqslant n$ на условие $0 \leqslant x_{ij} \leqslant 1$, $1 \leqslant i,j \leqslant n$, то получим задачу линейного программирования (*ЗЛП*). Оптимальное решение *ЗЛП* дает нижнюю оценку для решения TSP.

На вход TSP подается n^2 чисел (длина входа $O(n^2)$). Несмотря на то, что *ЗЛП* полиномиально разрешима [102], но в *ЗЛП* группа исключения подциклов (1.5) содержит ограничений $O(2^n)$. Таким образом, необходимо заменить условия (1.5) на другие условия, имеющие полиномиальное число ограничений (см. [5]). В этом случае получим полиномиально вычислимую нижнюю оценку TSP (1.4)-(1.5), которую можно получить, исключив ограничения (1.5) из решения задачи о назначениях (1.4), и улучшить оценку, последовательно добавляя ограничения на подциклы.

Соответствующий алгоритм может использоваться для решения TSP и $mTSP$.

Алгоритм 1.1. *Релаксационный алгоритм решения TSP и получения нижней оценки*

1: *Найти решение задачи о назначениях.*
2: *По заданному решению найти подмножество S, для которого нарушается условие на подциклы, и добавить их в систему ограничений.*
3: *Повторить процедуру решения, пока не найдено точное решение ЗЛП или пока задача не становится сложной для решения.*
4: *Найти нижнюю оценку.*

Линейная задача о назначениях полиномиально разрешима ([75; 102]). Справедливо утверждение

Утверждение 1.1. *Пусть для некоторого вектора* $\Delta = (\Delta_1, \ldots, \Delta_n)$ *существует набор* Δ*-минимальных элементов* $(c_{1j_1}, \ldots, c_{nj_i})$ *по одному в каждой строке и каждом столбце* ($c_{ij} - \Delta$ *— минимальный в строке* i*, если* $c_{ij} - \Delta_j \leqslant c_{ik} - \Delta_k$ *для всех* $k = \overline{1,n}$*). Тогда этот набор является оптимальным решением линейной задачи о назначениях.*

В агентном подходе возникает свой класс задач, связанных с поведением и управлением интеллектуальными агентами в условиях взаимодействия при решении прикладных экстремальных задач на графах, в частности, в работах автора [20; 23] представляются результаты сравнительного анализа алгоритмов для двух агентов-коммивояжеров, основанных на кластеризации графа.

Многоагентный подход к решению задачи коммивояжера — это обобщение хорошо известной *TSP*, где в решении участвуют несколько агентов-коммивояжеров. Данная задача комбинаторной оптимизации имеет множество применений, главным образом, в областях маршрутизации и планирования. Таких, как задача о школьном автобусе с ограничением на количество транспортных мест и количество учеников, задача о доставке. Нахождение эффективного алгоритма для решения многоагентной *TSP* является актуальной задачей.

Задача с несколькими агентами-коммивояжерами в общем случае может быть формализована следующим образом.

Дано $n > 1$ вершин и m агентов-коммивояжеров, расположенных в одной вершине-депо. Оставшиеся вершины, которые необходимо посетить, называются промежуточными вершинами. Таким образом, многоагентный подход к решению задачи включает нахождение маршрутов для всех m коммивояжеров, которые начинают и заканчивают свой путь в определенной вершине, при этом посещая каждую промежуточную вершину не более одного раза, минимизируя общую длину маршрута для всех агентов. При многоагентном подходе n вершин исходного графа должны быть разбиты на m маршрутов, на каждом из которых должна быть решена *TSP* с одним агентом.

Многоагентный подход к решению *TSP* при расположении всех агентов в одной вершине-депо. Пусть дан граф $G = (V,U)$, где V — множество вершин $V = 0,1,\ldots,n$, и U — множество дуг (ребер) и $C = (c_{ij})$ — матрица весов (расстояний), связанная с каждой дугой $(i,j) \in U$. Пусть m коммивояжеров расположены в вершине-депо $i = 0$. Многоагентный подход к решению *TSP* при расположении всех агентов в одной вершине-депо включает в себя нахождение всех маршрутов для m коммивояжеров таких, что они начинаются и заканчиваются в одной вершине. Все остальные вершины распределены по конкретным маршрутам. Количество вершин, посещаемых агентом, находится в пределах предопределенного интервала и общая стоимость посещения всех вершин минимизируется.

Пусть

$$x_{ij} = \begin{cases} 1, \text{если агент проходит по дуге } (i,j), \\ 0, \text{в противном случае;} \end{cases}$$

u_i — количество вершин, посещенных от источника до вершины i (т. е. номер посещения i-й вершины); L — максимальное количество вершин, которые коммивояжер может посе-

тить; K — минимальное количество вершин, которые коммивояжер должен посетить, то есть $K \leqslant u_i \leqslant L$.

Формализация многоагентной $mTSP$ в этом случае имеет вид:

$$\min \sum_{(i,j)\in U} c_{ij}x_{ij}, \tag{1.6}$$

$$\sum_{j=2} x_{1j} = m, \quad \sum_{j=2} x_{j1} = m, \tag{1.7}$$

$$\sum_{i=1} x_{ij} = 1, \quad j = 2,\ldots,n, \tag{1.8}$$

$$\sum_{j=1} x_{ij} = 1, \quad i = 2,\ldots,n, \tag{1.9}$$

$$u_i + (L-2)x_{1i} - x_{i1} \leqslant L-1, \quad i = 2,\ldots,n, \tag{1.10}$$

$$u_i + x_{1i} + (2-K)x_{i1} \geqslant 2, \quad i = 2,\ldots,n, \tag{1.11}$$

$$x_{1i} + x_{i1} \leqslant 1, \quad i = 2,\ldots,n, \tag{1.12}$$

$$u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leqslant L-1, \quad 2 \leqslant i \neq j \leqslant n, \tag{1.13}$$

$$x_{ij} \in \{0,1\}, \quad \forall (i,j) \in U. \tag{1.14}$$

$$f(x) = 1, \quad x = (x_{11}, x_{12}, \ldots, x_{nn}). \tag{1.15}$$

Целевая функция (1.6) минимизирует общее пройденное расстояние в маршруте. Условия (1.7) гарантируют, что m коммивояжеров начинают и заканчивают свой путь в одной вершине. Уравнения (1.8) и (1.9) являются ограничениями посещения вершин. Условия (1.10) и (1.11) накладывают ограничение на количество вершин, которые коммивояжер посетит (при $u_i = 1$, если i — первая вершина в маршруте). Ограничение (1.12) не позволяет коммивояжеру посещать только одну вершину. Неравенство (1.13) гарантирует, что $u_j = u_{i+1}$, тогда и только тогда, когда $x_{ij} = 1$. Таким образом, ограничения запрещают формирование каких-либо подмаршрутов между вершинами в $V\backslash\{1\}$ [203]. Условие (1.15) задает ограничения и дополнительные условия в виде *ДНФ*.

Многоагентный подход к решению TSP при расположении всех агентов в разных вершинах является обобщением рассматриваемой выше задачи, при котором у каждого агента разные вершины-депо.

Пусть дан граф $G = (V,U)$, где V — множество вершин $V = 0,1,\ldots,n$, и U — множество дуг $U = \{(i,j) : i,j \in V, i \neq j\}$, n — количество вершин. Множество всех вершин графа есть объединение $V = D \cup \widetilde{V}$, где D — это множество стартовых позиций агентов — депо. В узле i расположен коммивояжер m_i. И общее количество агентов — m. Пусть $\widetilde{V} = \{d+1, d+2, \ldots, n\}$ будет множеством клиентов, где $|D| = d$. C — матрица стоимостей переходов из одной вершины в другую графа G, которая обладает свойствами: $c_{ij} = c_{ji}$ и $c_{ij} + c_{jk} \leqslant c_{ik}$ для каждого $i,j,k = 1,2,\ldots,n$.

Пусть x_{ij}, L, K определены, как и раньше. Тогда получим следующую формализацию многоагентной TSP:

$$\min \sum_{(i,j)\in U} c_{ij}x_{ij}, \tag{1.16}$$

$$\sum_{j\in\widetilde{V}} x_{ij} = m_i, \quad i \in D, \tag{1.17}$$

$$\sum_{i\in\widetilde{V}} x_{ij} = m_j, \quad j \in D, \tag{1.18}$$

$$\sum_{i\in V} x_{ij} = 1, \quad j \in \widetilde{V}, \tag{1.19}$$

$$\sum_{j\in V} x_{ij} = 1, \quad i \in \widetilde{V}, \tag{1.20}$$

$$u_i + (L-2)\sum_{k\in D} x_{ki} - \sum_{k\in D} x_{ik} \leqslant L-1, \quad i \in \widetilde{V}, \tag{1.21}$$

$$u_i + \sum_{k\in D} x_{ki} + (2-L)\sum_{k\in D} x_{ik} \geqslant 2, \quad i \in \widetilde{V}, \tag{1.22}$$

$$x_{ki} + x_{ik} \leqslant 1, \quad k \in D, \quad i \in \widetilde{V}, \tag{1.23}$$

$$u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leqslant L-1, \quad i \neq j; \quad i,j \in \widetilde{V}, \tag{1.24}$$

$$x_{ij} \in \{0,1\}, \quad i,j \in V. \tag{1.25}$$

$$f(x) = 1, \quad x = (x_{11}, x_{12}, \ldots, x_{nn}) - \textit{ДНФ} \text{ ограничения}. \tag{1.26}$$

Целевая функция (1.16) минимизирует общее пройденное расстояние в маршруте. В данной формулировке для каждого $i \in D$ уравнения (1.17) и (1.18) гарантируют, что m_i коммивояжер начинает путь в вершине i. Уравнения (1.19) и (1.20) являются ограничениями посещения вершин. Условия (1.21) и (1.22) накладывают ограничения на количество вершин, которые коммивояжер посетит при $u_i = 1$, если i — это первая вершина в маршруте. Ограничение (1.23) не позволяет коммивояжеру посещать только одну вершину. Ограничение (1.24) разбивает все подмаршруты между агентами [203].

В постановке задачи $mTSP$ (1.6)–(1.15) и (1.16)–(1.26) агенты-коммивояжеры не конкурируют друг с другом, а обеспечивают минимальный по стоимости (расстоянию) маршрут. Данная модель позволяет добавлять ограничения в виде *ДНФ* [18]. Добавление ограничений и учет априорных знаний в виде *ДНФ* позволяют уточнить псевдобулевую постановку задач (1.6)–(1.26). Далее рассмотренным моделям ставятся в соответствие модели с *ДНФ* ограничениями, которые полиномиально разрешимы.

Полученная задача псевдобулевой условной оптимизации удобна для теоретического анализа, проведения декомпозиции по характерным блокам, а также позволяет выделять задачу о назначении с дополнительными условиями (см. раздел 1.2.1). В случае конкуренции агентов, вместо критерия (1.6) будет m критериев, отвечающих каждому агенту. В многокритериальной задачи псевдобулевой оптимизации необходимо найти Парето-оптимальные

решения (эффективные), т. е. решение в этом случае есть результат компромисса. Более естественной является игровая постановка задачи m коммивояжеров [20].

В ряде задач требуется проходить маршрут по уже выделенным ранее последовательностям дуг или обязательно обходить кластеры. Такие задачи требуют уточнения постановок (1.6)–(1.26).

В рамках модели псевдобулевой оптимизации естественно использовать указанные задачи в разработке алгоритмов маршрутизации многих коммивояжеров.

Рассмотрим задачу определения количества агентов-коммивояжера. Утверждение: задача определения необходимого количества агентов-коммивояжеров имеет полиномиальную сложность. Для сложной сети необходимо выяснить, сколько нужно агентов-коммивояжеров для совместного обхода вершин сети непересекающимися маршрутами коммивояжеров. Поставим данной задаче модель задачи о назначениях (1.4), решение имеет полиномиальную сложность. В результате получаем несколько кластеров, которым соответствуют маршруты коммивояжера. Если количество маршрутов k соответствует предполагаемому количеству агентов m, то решение задачи заканчивается. В противном случае необходимо объединить маршруты ($k > m$) или разбить на несколько ($k < m$). Понятно, что такой подход далеко не всегда приводит к искомому решению, но оценки маршрутов, полученные с помощью задачи о назначениях, служат оценками для ветвления в соответствующей задаче коммивояжера на кластере.

1.3 Управление в многоагентных системах

1.3.1 Интеллектуальное управление агентами в системах с локальной организацией

Многоагентный подход к задачам маршрутизации типа $mTSP$ в сложных сетях является частью многоагентной системы (MAC) интеллектуального управления для локально организованных систем. Рассмотрим некоторые аспекты управления агентами.

Локальная организация вместе с централизованной и децентрализованной организацией присущи большинству реальных сложных социально-экономических и технических систем. В работах В. Л. Стефанюка [125] методология локально-организованных систем последовательно реализована на классах задач, имеющих научное и прикладное значение: коллективное поведение, локальное управление, локальное программирование, локально организованные системы искусственного интеллекта ($ИИ$), локально организованные экспертные системы. По мнению автора, такой подход представляет перспективное направление в теории и практике исследования и создания больших систем.

Локальность требует разработки соответствующих инструментов, допускающих инсталляцию для конкретных больших систем. Наиболее адекватным для широкого класса задач

на локально организованных структурах являет подход, базирующийся на интеллектуальных агентах (*ИА*). Характерным примером такого типа является задача интеллектуального управления агентами-коммивояжерами, приведенная в работах автора [19; 74].

По интеллектуальному управлению, как правило, представлены результаты, относящиеся к частным разделам и задачам. Отметим обобщающий обзор В. И. Донского [51], в котором сделаны выводы о важности и перспективности такого направления.

Интеллектуальное управление представляется одной из актуальных областей применения теоретической кибернетики в современных информационных системах. Отметим, что важными направлениями развития теории интеллектуального управления являются: разработка специфических моделей памяти для агентов, алгоритмических баз знаний агентов, моделей динамической автоматической классификации состояний управляемых объектов и среды, автоматического изменения структур математических моделей и даже автоматической смены используемой агентом модели.

Под *ИА* обычно понимают программы, которым человек передает свои функции с целью автоматического (полуавтоматического) принятия решений. Агенты запоминают нужную для правильного функционирования информацию, обобщают ее, обучаются и могут самостоятельно генерировать управляющие воздействия или выдавать полезные рекомендации пользователям [13; 14; 20; 21; 23; 29; 30; 36].

Под автономностью *ИА* понимают их способность контролировать самих себя, реализовывать самоуправление и целевое поведение: агенты имеют цель и действуют в соответствии с этой целью. Агент — это объект, погруженный в среду, от которой он получает информацию, отражающую события, происходящие в этой среде; интерпретирует события и исполняет команды, воздействующие на среду. Агент может содержать программные и аппаратные компоненты [121]. Под *ИА* понимается программно или аппаратно реализованная система, обладающая такими свойствами:

— автономность — способность функционировать без вмешательства человека и осуществлять самоконтроль над своими действиями и внутренним состоянием;

— способность функционировать в сообществе с другими агентами, обмениваясь с ними сообщениями с помощью некоторого специального языка коммуникаций;

— реактивность — способность воспринимать состояние среды и своевременно реагировать на ее изменения;

— способность генерировать цели и действовать, стремясь достигнуть их;

— обладать знаниями о себе, среде и, возможно, других агентах; пополнять и модифицировать их.

Из приведенного определения *ИА* видно, что управляющие функции для них являются основными, и причин рассматривать *ИА* вне парадигмы интеллектуального управления нет.

Заметим, что в каждом конкретном случае понадобятся многочисленные уточнения, главное из которых будет определяться требуемой степенью автономности интеллектуальной управляющей системы.

Исторически наиболее ранние и важные результаты по моделированию поведения агентов в многоагентных системах, т. е. организации их индивидуальных предпочтений и

взаимодействия с другими агентами, получены в теории коллективного поведения агентов и игр автоматов, развиваемой школой М. Л. Цейтлина [125, с. 9].

Ключевое слово «моделирование» относится ко всей цепочке задач интеллектуального управления. Это моделирование локальной структуры системы, иерархичности системы управления, системы глобальных и локальных целей по уровням системы и для агентов (синтез целей). При этом интеллектуальное управление отражает методы управления, свойственные человеку с использованием адаптации, обучения, прогнозирования и предвидения в условиях неопределенности и большого объема обрабатываемой информации. Агенты, реализующие интеллектуальное управление, используют прецеденты, экспертные знания, прототипы, модели агентов, встроенные в нейросетевое управление и включающие различные системы вывода. Как знаниеориентированные интеллектуальные системы они требуют всестороннего использования различного рода информации (знаний).

Для агентов-коммивояжеров проявляется специфика задачи в существовании двух процессов: это процесс декомпозиции исходной задачи с помощью m коммивояжеров (кластеризации) и интеграционный процесс локального поведения коммивояжеров. Из-за NP-полноты задач типа коммивояжера агенты-коммивояжеры используют совокупность эвристик — как разновидность локального подхода к решению задач. Здесь важным является выбор удобного представления для решаемой задачи (принцип семиотической интроспекции, т. е. механизм для определения аналогичности (подобия), который не решает задачу, но может применяться для продвижения к подобной цели [125, с. 157]).

В [51] приводится пример *ИА*, управляющего клеточной игрой, и пример реализации интеллектуального управления трафиками в транспортной сети с пересечениями. В [125] описано исследование задачи локального управления потоками такси, а также результаты имитационного моделирования для решения проблем заторов.

Исходя из сочетания локального и глобального характера знаний о задаче типа коммивояжера в работах [26; 125] анонсирован подход по рациональному рассмотрению многокритериальных задач нескольких коммивояжеров, как агентных в самоорганизующейся системе.

Рассмотрим несколько предлагаемых новых постановок задач $mTSP$:

1. На уровне системы интересы (цели, критерии) коммивояжеров-агентов общие, локальные интересы отдельного агента представлены локальными целями, работающие на общую цель (которую агенты могут и не осознавать). При этом локальные цели, критерии, ограничения и ресурсы общедоступны для всех агентов. Оптимальность многокритериального решения достигается за счет обмена информацией (знаниями) между агентами-коммивояжерами с помощью процедуры самоорганизации или некоторой другой.
2. Каждый агент руководствуется индивидуальной частью базы целей и критериев, а достижение общей цели (оптимального решения) согласовывается между агентами (процесс согласования — самоорганизации является параллельным процессом для агентов в рамках большой системы — сетевой структуры с ограничениями заданными в базе знаний).

3. Агенты-коммивояжеры конкурируют между собой, захватывая часть сети в соответствии со своими целями и критериями. Формализация в этих случаях приводит к различным постановкам игровых многокритериальных задач для многих коммивояжеров-агентов (в частности, игры автоматов).
4. Агентная задача многих коммивояжеров на изменяющейся сети (в условиях чрезвычайных событий) естественно возникает, когда в рамках большой сетевой системы реализуется оптимальная сеть (синтез сети). На базе известных решений проводится реоптимизация относительно добавления вершин, дуг, внутренних условий (*ДНФ* ограничений), знаний, по количеству коммивояжеров-агентов. Устойчивость задач или оптимальных решений позволяет строить эффективные алгоритмы.

Заметим, что количество агентов-коммивояжеров связано с декомпозицией исходной задачи большой размерности, с выделением характерных кластеров (город, район и т. п.). С другой стороны решение задачи кластеризации может осуществляться с помощью агентов-коммивояжеров (задача распознавания конечных графов несколькими агентами [124]). Отметим важность и актуальность агентного подхода в интеллектуальном управлении.

В зависимости от класса решаемых задач многоагентные интеллектуальные системы (*МАИС*) могут иметь различную структуру, состоять из однотипных *ИА* или специализированных агентов, которые находятся под управлением интеллектуальной системы (внешний по отношению к агентам) или управляются другими *ИА*. Интеллектуальность может содержаться во внешней интеллектуальной системе управления (*ИСУ*), распределенной по *ИА* или гибридной с адаптируемой к внешней среде структурой. Интеллектуальные системы (*ИС*) характеризуются способностью извлекать, хранить и обрабатывать знания, что обеспечивает функционирование *ИС* при управлении сложными объектами (системами). *ИСУ* должны обладать способностью к обучению, адаптации к внешней среде, живучестью (устойчивостью к помехам и повреждениям), функционировать совместно с лицом, принимающим решение (*ЛПР*). Такие системы могут быть самоорганизующимися и обучающимися. Кроме указанных, *МАИС* могут обладать и другими свойствами.

Будем предполагать, что возможности *ИА*, как и *МАИС* в целом, изменяются в процессе достижения цели. Обучаясь и адаптируясь к изменяющимся условиям, ориентируясь на априорную, прецедентную информацию и полученные *ИА* локализованные решения *МАИС*, расширяет (или сужает) область поиска решений и увеличивает скорость оптимизации.

Важно, что модели интеллектуального управления в многоагентной системе *ИА* используются для решения задач дискретной оптимизации: TSP, $mTSP$ и др. Отметим, что условиям адаптивности отвечают генетические алгоритмы (GA) — одни из наиболее эффективных эвристических алгоритмов для *МАИС*. Пусть *ИА* обладает набором метаэвристических алгоритмов и возможностью их сочетания, и распараллеливания. Для определенности в качестве модели *ИСУ* выберем модель, основанную на применении GA. В такой интерпретации множество *ИА* образует популяцию, которая разбивается на несколько самостоятельных подпопуляций, каждая из которых использует свой набор алгоритмов. Самостоятельные подпопуляции могут работать параллельно, на выделенных кластерах, в соответствии с декомпозицией исходной задачи и т. п.

Для взаимодействия может быть выделена подпопуляция *ИА* (или отдельный *ИА*). В *МАИС* осуществляется обмен информацией (решениями, алгоритмами и т. д.). Обмен генетической информацией между подпопуляциями приводит к совершенствованию *ИА*. Если рассматривать дискретные моменты времени, то каждому следующему моменту времени соответствует своя *МАИС*, полученная в результате генетических преобразований. Структура подпопуляции *ИА* и их взаимодействие, кроме генетических структур, порождает искусственную нейронную сеть и систему вывода.

Таким образом, уже в процессе проектирования *МАИС* на базе *GA* решается ряд задач:

— идентификация системы целей, критериев оценки сложности изучаемой системы (задачи) и возможности декомпозиции (кластеризации, распараллеливания);

— разработка моделей взаимодействия популяции *ИА*, входящих в состав *МАИС*;

— разработка изменений *ИА* в подпопуляциях на базе обмена генетической информацией между подпопуляциями и получаемых знаний;

— разработка условий забывания информации в *ИА*;

— выяснение условий перехода *ИА* из одной в другую подпопуляцию;

— выяснения условий эволюции структуры популяции *МАИС* и устойчивости.

Новое поколение автономных взаимодействующих агентов может создавать *МАС*, а именно: популяцию размножающихся агентов. В работе В. В. Стальского [123] рассматривается эволюционных подход когнитивного развития обучающихся автономных агентов.

Предложенный в [123] метод эволюционного развития «размножающихся агентов» и их популяции, основан на том, что далеко продвинутому «обучающемуся рефлексному автономному агенту» [105; 106] придана функция «самовоспроизведения». В соответствии с предлагаемым методом можно построить модель, в которой обучающиеся рефлексные агенты (артифакты) имитируют размножение организмов в живой природе. Каждый агент работает в автоматическом режиме и следует целевой функции, имитирующей инстинкт полового размножения высших животных и человека. Целевая функция (половой инстинкт) предопределяет неформальное общение агента с другими агентами, стремление агента опередить конкурентов, т. е. преуспеть в обществе себе подобных (в популяции агентов), которая по определению является совокупностью индивидов, характеризующаяся общностью происхождения и образующая целостную генетическую систему. В результате этого в популяции агентов реализуется «естественный отбор», и происходит повышение когнитивного уровня всей популяции от поколения к поколению.

Отметим, что важным преимуществом такого агента-«специалиста» является возможность передачи полученных им знаний в полном объеме «потомству». Потомок будет владеть не только опытом, основанным на прежних решениях, но и опытом работы логических и вычислительных алгоритмов, т. е. всем ходом «размышлений» предыдущих поколений агентов. Таким образом, может быть создана «династия» специальных систем *ИИ*, например, экспертных [106]. Непрерывное совершенствование особей, составляющих популяцию, обеспечивает ее эволюцию, т. е. система (структура) совершенствуется за счет ее элементов. При этом будет существовать и обратная связь: развитая эволюционирующая популяция позитивно влияет на развитие отдельных индивидов.

В работе [123] обоснованы формулы, обеспечивающие контроль процессов размножения индивидов и контроль развития популяции в целом. Создание подобной искусственной популяции согласуется с современными тенденциями в науке, например, в работах [105; 127] приведена позиция антропоморфного взгляда на сообщество агентов (она рассматривается под углом зрения искусственной жизни).

Близкие к [123] результаты содержатся также в работе А. В. Смирнова, Л. Б. Шеремета [122], где дан обзор по построению моделей формирования коалиций между рациональными кооперативными агентами. В [122] рассмотрены основные подходы к построению многоагентных коалиционных систем, в которых алгоритмы формирования коалиций являются способом самоорганизации заинтересованных агентов с целью адаптации к изменениям, как окружающей среды, так и самой системы. Показано, что формирование коалиций — это способ создавать виртуальные организации агентов посредством применения согласованных стратегий, временная логика которых зависит от динамически меняющихся условий. Отметим, что задача формирования коалиций в сетевой постановке приводит к кооперативным играм на графах.

В работе М. А. Михеенковой, В. К. Финна [98] дана формализация поведения искусственного агента и группы агентов на основе структурированного представления агента и его действий; рассматриваются *ДСМ*-метод, семантика логики аргументации и критерии рациональности действий, позволяющие решать задачи анализа и предсказания действий агентов и рациональности их поведения, а также выбирать решения, увеличивающие степень рациональности как одного, так и системы агентов. Предлагаемый в [98] подход позволяет делать не только анализ и предсказание поведения отдельного агента, но и формировать классы агентов на основе доминант набора действий.

Далее в работе (глава 2) будут приведены модели в виде псевдобулевой оптимизации с *ДНФ* ограничениями, которые служат теоретической основой для MAC типа $mTSP$ и систем управления.

Для сложных сетей рассмотренные подходы по использованию многоагентных систем являются наиболее предпочтительными. Среди множества возможных приложений выделим многоагентный подход в задачах маршрутизации и задачах многих коммивояжеров с дополнительными ограничениями. Разработка программных продуктов, основанных на метаэвристиках, востребована в задачах безопасности, организации компьютерных сетей, поведении коллектива мобильных роботов (дронов). Модельной может быть задача для двух коммивояжеров, осуществляющих взаимодействие (обмен знаниями) в процессе решения указанных оптимизационных сетевых задач.

Выводы

1. Дано определение знаниеориентированных задач маршрутизации.

2. Очерчен круг одно- и многоагентных задач маршрутизации типа коммивояжеров (TSP, $mTSP$), моделирующих практические прикладные задачи. Показана важность учета знаний о решениях, компонентах моделей, структуре сложных сетей для разработки перспективных алгоритмов маршрутизации.

3. Показано, что задачи многоагентной маршрутизации, как правило, являются NP-полными, при этом выделяются классы задач, для нахождения решений которых существуют полиномиальные алгоритмы. Но сведение исходной задачи к полиномиально разрешимой, в свою очередь, может потребовать решения NP-полной задачи.

Необходим компромисс, который достигается в сведении исходной задачи к задаче меньшей размерности, но NP-сложной, или выбор схемы сведения к полиномиально разрешимым задачам, когда приближенное решение схемы может быть осуществимым. Такой подход требует предъявления набора обобщенных моделей, задач, алгоритмов, прецедентных решений, который позволяет синтезировать композицию алгоритмов (модель) для построения $mTSP$.

Методология разработки алгоритма решения задач маршрутизации может быть основана на формировании по исходной сложной сети более простой (относительно реализации алгоритмов маршрутизации) по своей структуре сети.

4. Рассмотрен вопрос интеллектуального управления агентами в MAC. Перспективным является подход, в котором агенты-коммивояжеры рассматриваются как популяция в рамках модели GA.

Дальнейшие исследования связаны с обучением агентов-коммивояжеров, их автономностью и организацией обмена прецедентной информацией между агентами (системами управления).

Глава 2. Знаниеориентированные модели и методы решения многоагентной задачи коммивояжера

2.1 Модели псевдобулевой оптимизации с дизъюнктивными ограничениями

Прикладная теория задач маршрутизации на сложных сетях (типа многих агентов-коммивояжеров) базируется на точных решениях выделенных классов задач с полиномиальными алгоритмами решения, на использовании приближенных алгоритмов решения (например, с гарантированной функциональностью) и декомпозиции (кластеризации) исходной задачи, т. е. сведении к задачам меньшей размерности и уточняющих преобразованиях для возврата к исходной задаче. Ещё раз отметим, что важным в этом процессе является учет всей имеющейся информации, знаний, фактов и прецедентов, как для построения иерархии моделей (извлечение моделей), так и для разработки практических алгоритмов решения [6; 32; 50; 52; 70—72; 92; 173; 174].

Разнообразие алгоритмов решения таких задач связано с наличием априорных знаний о решении или структуре сети, прецедентным характером знаний и требованиями к точности решения. Рациональным является использование, как точных, так и приближенных алгоритмов и их композиций.

Многоагентные системы (в частности с роевым интеллектом) используются для решения сложных задач дискретной оптимизации, которые нельзя эффективно решать классическими алгоритмами. Как указывалось в разделе 1.3, агентная модель для сложной сети задачи типа $mTSP$ становится интеллектуализированной системой, определяющей эвристические алгоритмы поиска оптимального решения реактивными агентами (исполняющих заложенные в них правилах).

Синтез *МАС ИИ* по частичной, прецедентной, априорной информации базируется на результатах наблюдения над поведением *МАС* на основе накопленной информации в виде [173]: «... вектора состояния, значения качества функционирования системы, бинарного индикатора допустимости этого состояния». Для *МАС* маршрутизации типа $mTSP$ в работе используется модель скалярной псевдобулевой условной оптимизации с ограничениями в виде *ДНФ*. Такие модели естественным образом учитывают линейные ограничения по прохождению вершин сети, декларативные требования, а также требования предшествования, обязательного прохождения выделенного множества дуг и другую прецедентную информацию.

Псевдобулевые оптимизационные модели с сепарабельными целевыми функциями и *ДНФ* ограничениями, имеющими ограниченную постоянной длину, являются полиномиально разрешимыми. Представляют интерес классы задач, которые приведены или легко приводятся к форме с *ДНФ* ограничениями, так как в общем случае такие приведения являются экспоненциальными. Синтез модели с *ДНФ* ограничениями из данных можно осуществлять приближенно, и сложность такой аппроксимации оказывается полиномиальной.

В работе [173] показано, что число конъюнкций в извлеченной *ДНФ* не превышает числа примеров в исходной прецедентной информации. При этом указывается, что для построения *ДНФ* ограничений целесообразно использовать решающие деревья. В случае монотонности и линейности частично заданной целевой функции в работах В. И. Донского [52; 173] и М. Г. Козловой [70; 71] предложены алгоритмы решения задач псевдобулевой скалярной оптимизации при наличии неполной, прецедентной начальной информации. Методология такого подхода будет применена для решения многоагентных задач типа многих коммивояжеров. Результат данной главы представлен в декабре 2020 года на Международной конференции «Интеллектуализация обработки информации» [32].

В данной главе выделено важное направление — построение $MAC\ mTSP$ на базе моделей псевдобулевой оптимизации с дизъюнктивными ограничениями.

2.1.1 Задачи псевдобулевой оптимизации

В связи с задачами маршрутизации на графах особый интерес представляют задачи псевдобулевой оптимизации (с булевыми переменными), которые имеют широкие приложения [6; 52; 168]. Достаточно подробно такие задачи исследовались в работах [180; 190], где разработаны методы решения в случае аналитически заданных моделей псевдобулевой оптимизации. В работе [163] псевдобулевые функции рассматриваются, как отображения из семейства подмножеств конечного исходного множества действительных чисел. Оптимизация на графах в классе псевдобулевых функций представлена в работах [176; 192]. Базовые результаты содержатся в [59; 60].

Введем следующие обозначения:

$B^n = \{0,1\}^n$ — единичный n-мерный куб, $P_2(n) = \{F : B^n \to \{0,1\}\}$ — класс функций алгебры логики (*ФАЛ*), зависящих от n переменных, $\tilde{x} = (x_1, x_2, \ldots, x_n) \in B^n$.

Функция вида $f : B^n \to \mathbb{R}$, где $\mathbb{R}$ — множество действительных чисел, называется псевдобулевой [6; 52; 163]. Обозначим через $PS_2(n)$ множество псевдобулевых функций, а через $LPS_2(n)$ — множество линейных псевдобулевых функций $PS_2(n)$.

Задача вида

$$extr f(\tilde{x}), \quad \tilde{x} \in \Omega \subseteq B^n, \quad f \in PS_2(n) \tag{2.1}$$

называется задачей псевдобулевой оптимизации.

Введем характеристическую функцию множества ограничений Ω:

$$F_\Omega(\tilde{x}) = \begin{cases} 1, & \tilde{x} \in \Omega; \\ 0, & \tilde{x} \in B^n \backslash \Omega. \end{cases}$$

и запишем задачу (2.1) в эквивалентной форме:

$$extr\ f(\tilde{x}), \quad F_\Omega(\tilde{x}) = 1, \quad f \in PS_2(n), \quad F_\Omega \in P_2(n), \quad \tilde{x} \in B^n, \tag{2.2}$$

где $P_2(n)$ — класс функций алгебры логики от n переменных.

Через $D_{F_\Omega} = \bigvee\limits_{j=1}^{m} K_j$ — обозначим произвольную дизъюнктивную нормальную форму функции $F_\Omega(\tilde{x})$, где K_j — элементная конъюнкция. Тогда задачу (2.2) можно записать:

$$extr\ f(\tilde{x}), \quad D_{F_\Omega}(\tilde{x}) = 1, \quad \tilde{x} \in B^n. \tag{2.3}$$

Задача (2.3) называется задачей псевдобулевой оптимизации с дизъюнктивным ограничением, а форму ее представления называют канонической.

Определение 2.1 ([137]). *Переменная x_i называется существенной для $f \in PS_2(n)$, если найдется такой набор значений переменных $\alpha_1,\ldots,\alpha_{i-1},\alpha_{i+1},\ldots,\alpha_n$, что $f(\alpha_1,\ldots,\alpha_{i-1},0,\alpha_{i+1},\ldots,\alpha_n) \neq f(\alpha_1,\ldots,\alpha_{i-1},1,\alpha_{i+1},\ldots,\alpha_n)$. В противном случае переменная называется фиктивной.*

Определение 2.2 ([137]). *Псевдобулевые функции f_1 и f_2 называются равными, если функция f_2 может быть получена из f_1 путем введения или удаления фиктивных переменных.*

В канонической форме псевдобулевую функцию f можно записать в виде [50]:

$$f(x_1,\ldots,x_n) = \sum_{\tilde{\sigma}\in B^n} a_{\tilde{\sigma}} x_1^{\sigma_1} \cdot \ldots \cdot x_i^{\sigma_i} \cdot \ldots \cdot x_n^{\sigma_n}, \tag{2.4}$$

где $\tilde{\sigma} = (\sigma_1,\ldots,\sigma_n)$, $\sigma_i \in \{0,1\}$, $i = \overline{1,n}$, $a_{\tilde{\sigma}} \in \mathbb{R}$, $x^\sigma = \begin{cases} x, & \sigma = 1, \\ \overline{x}, & \sigma = 0. \end{cases}$

Каждая псевдобулевая функция может быть представлена в полиномиальной форме над полем действительных чисел

$$f(x_1,\ldots,x_n) = \sum_{j=1}^{k_f} c_j x_{j_1} \cdot \ldots \cdot x_{j_{r_j}} + c_0, \quad c_j \in \mathbb{R}, \quad j = 1,\ldots,k_f. \tag{2.5}$$

Задача псевдобулевой оптимизации в форме слабых неравенств имеет вид:

$$\begin{gathered} extr\ f(x_1,\ldots,x_n), \quad g_j(x_1,\ldots,x_n) \leqslant 0, \quad j = 1,2,\ldots,m, \\ (x_1,\ldots,x_n) \in B^n, \quad f,\ g_j \in PS_2(n). \end{gathered} \tag{2.6}$$

Определение 2.3. *Две формы представления оптимизационной задачи называются эквивалентными, если множества их решений совпадают.*

Теорема 2.1 ([50]). *Для любой задачи псевдобулевой оптимизации в форме* (2.1) *существует эквивалентная форма представления*

$$extr\ f(\tilde{x}), \quad h(\tilde{x}) \leqslant 0, \quad \tilde{x} \in B^n \tag{2.7}$$

с единственным ограничением в виде нестрогого неравенства, где f,$h \in PS_2(n)$ есть некоторые полиномы.

Полиномиальное представление для функции $f \in PS_2(n)$ существует всегда.

Теорема 2.2 ([50]). *Любая задача оптимизации псевдобулевой функции с ограничениями, определяющими непустое множество допустимых решений* Ω *задачи* (2.1)*, может быть представлена в эквивалентной форме с дизъюнктивным условием* (2.3)*.*

Доказательство этой теоремы следует из существования эквивалентной формы задачи с характеристическими функциями на наборе допустимых значений $F_\Omega(\tilde{x})$ и полноты представления функций алгебры логики в виде дизъюнктивных нормальных форм.

Определение 2.4. *Представление задач произвольного класса* Z *в форме* F *называется полным в* Z*, если любая задача этого класса может быть представлена в форме* F*.*

Следствие 2.1. *Представление задач условной оптимизации псевдобулевой функции в форме с дизъюнктивным ограничением является полным.*

Область допустимых решений задачи (2.3) и эквивалентной ей задачи (2.1) может быть представлена в виде: $\Omega = \bigcup\limits_{j=1}^{m} N_{K_j}$, где N_{K_j} — интервал ранга r_j, соответствующий элементарной конъюнкции $K_j = x_{j_1}^{\sigma_{j_1}} \& \ldots \& x_{j_{r_j}}^{\sigma_{j_{r_j}}}$, что приводит к еще одной эквивалентной форме задачи (2.1):

$$\underset{1\leqslant j\leqslant m}{extr} \quad \underset{\tilde{x}\in N_{K_j}}{extr} \quad f(\tilde{x}). \tag{2.8}$$

Так как область допустимых решений есть объединение интервалов N_{K_j}, $j = \overline{1,m}$, легко убедиться, что экстремальное решение задачи можно определить путем его выбора из предварительно найденных допустимых решений, являющихся экстремальными в интервалах N_{K_j}.

Рассмотрим основные алгоритмы решения задач псевдобулевой оптимизации. Пусть дана задача псевдобулевой оптимизации с линейной целевой функцией

$$\max \sum_{i=1}^{n} c_i x_i, \quad \bigvee_{j=1}^{m} K_j(\tilde{x}) = 1, \quad \tilde{x} \in B^n, \tag{2.9}$$

где $K_j(\tilde{x}) = x_{j_1}^{\sigma_{j_1}} \& \ldots \& x_{j_{r_j}}^{\sigma_{j_{r_j}}}$, $j = \overline{1,m}$. Ее можно записать в эквивалентной форме:

$$\max \sum_{i=1}^{n} c_i x_i, \qquad \tilde{x} \in \bigcup_{j=1}^{m} N_{K_j}, \tag{2.10}$$

где N_{K_j} — интервал в B^n, соответствующий конъюнкции K_j; интервал N_{K_j} определяется набором значений $\{j_1, \ldots, j_{r_j}\}$ (направление) и множеством $\{\sigma_{j_1}, \ldots, \sigma_{j_{r_j}}\}$ (код интервала).

Решение (2.10) сводится к решению задачи

$$\max_{1\leqslant j\leqslant m} \max_{\tilde{x}\in N_{K_j}} \sum_{i=1}^{n} c_i x_i, \tag{2.11}$$

которое, в свою очередь, требует решения m задач вида:

$$\max_{\tilde{x}} \sum_{i=1}^{n} c_i x_i, \quad \tilde{x} : \ (x_{j_1} = \sigma_{j_1}) \& \ldots \& (x_{j_{r_j}} = \sigma_{j_{r_j}}). \tag{2.12}$$

Допустимыми решениями задачи (2.12) являются только те булевы наборы $\tilde{x}$, у которых зафиксированы координаты $x_{j_1}{=}\sigma_{j_1},\ldots,\ x_{j_{r_j}}{=}\sigma_{j_{r_j}}$, а остальные принимают любые значения из множества $\{0,1\}$. Свободные переменные (вне множества номеров $\{j_1,\ldots,j_{r_j}\}$) можно назначать единичными или нулевыми в зависимости от значения $c_i, i \in \{1,2,\ldots,n\}\backslash\{j_1,\ldots,j_{r_j}\}$ — коэффициентов целевой функции. Экстремальные решения $\tilde{x}^*$ задачи (2.12) будут определяться формулой [191]:

$$x_i^* = \begin{cases} \sigma_i, & i \in \{j_1,\ldots,j_{r_j}\}, \\ \varphi(c_i), & i \notin \{j_1,\ldots,j_{r_j}\}, \end{cases} \tag{2.13}$$

где

$$\varphi(c_i) = \begin{cases} 1, & c_i > 0, \\ 0, & c_i < 0, \\ \alpha, & c_i = 0, \end{cases}$$

α — любое значение из $\{0,1\}$. В случае, когда все $c_i \neq 0$, задача (2.12) имеет единственное решение, а исходная задача (2.11) — не более m решений.

На каждом интервале N_{K_j} при вычислении x_i^*, согласно (2.13), просматривается n значений, а интервалов всего m, поэтому сложность решения $O(mn)$.

Теорема 2.3 ([50]). *Если задача условной оптимизации линейной псевдобулевой функции с ограничениями-неравенствами приводится к эквивалентной форме с дизъюнктивным ограничением за число шагов, ограниченное полиномом от размерности задачи, то она разрешима за полиномиальное время.*

2.1.2 Псевдобулевая модель задачи коммивояжера

Рассмотрим *TSP*, формализованную в виде модели линейной псевдобулевой условной оптимизации с неотрицательными коэффициентами ($c_{ij} \geqslant 0$) целевой функции

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min, \tag{2.14}$$

$$\sum_{i=1}^{n} x_{ij} = 1,\ \sum_{j=1}^{n} x_{ij} = 1,\ i,j = \overline{1,n}, \tag{2.15}$$

$$u_i - u_j + nx_{ij} \leqslant n-1,\ i,j = \overline{2,n},\ i \neq j. \tag{2.16}$$

Здесь u_i — произвольные действительные числа (в частности, им может соответствовать нумерация вершин, по которым проходит коммивояжер). Ограничения (2.16) препятствуют образованию подциклов. Для того, чтобы упростить выкладки, будем использовать обозначение двухиндексных величин через одноиндексные: $\tilde{x} = (x_1,\ x_2,\ldots,\ x_N) = (x_{11},\ x_{12},\ldots,\ x_{nn}) \in B^N$, $c = (c_1,\ c_2,\ldots,\ c_N) = (c_{11},\ c_{12},\ldots,\ c_{nn})$, где $N = n^2$. Ограничениям можно поставить в соответствие функции $F_j(\tilde{x}) \in P_2(N)$, $j = \overline{1,M}$, где M — число ограничений. Можно в $\tilde{x}$ использовать и другую нумерацию элементов: $x_{ij},\ i,j = \overline{1,n}$.

Определение 2.5 ([109]). *Вершина $\tilde{\alpha}$ куба B^n называется верхним нулем монотонной функции $f(\tilde{x})$, если $f(\tilde{\alpha})=0$, и для всякой вершины $\tilde{\beta}$ из $\tilde{\alpha} \prec \tilde{\beta}$ следует, что $f(\tilde{\beta})=1$.*

Лемма 2.1. *Если*

$$\left\{\sum_{j=1}^{n} x_{ij}=1,\ \sum_{i=1}^{n} x_{ij}=1,\ i,j=\overline{1,n},\ u_i-u_j+nx_{ij}\leqslant n-1,\quad i\neq j\right\}\Leftrightarrow$$

$$\Leftrightarrow \{F_j(\tilde{x})=0\},$$

то F_j — монотонные ФАЛ и задачу (2.14)–(2.16) *можно записать в виде*

$$(c,\tilde{x})\to\min,\ F_0(\tilde{x})=\bigvee_{j=1}^{M} F_j(\tilde{x})=0, \tag{2.17}$$

где $c=(c_{11},c_{12},\ldots,c_{nn})\equiv(c_1,c_2,\ldots,c_N)$, $\tilde{x}=(x_{11},x_{12},\ldots,x_{nn})\equiv(x_1,x_2,\ldots,c_N)$ $(c,\tilde{x})$ — скалярное произведение $\left((c,\tilde{x})=\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}\right)$, $F_0(\tilde{x})$ — монотонная ФАЛ.

Доказательство. Покажем монотонность функций $F_j(\tilde{x})$. Пусть $\tilde{\alpha}$, $\tilde{\beta}\in B^N$ такие, что $\tilde{\alpha}\preccurlyeq\tilde{\beta}$, т. е. $\alpha_{ij}\leqslant\beta_{ij},\ \forall i,j=\overline{1,n}$. Тогда

$$S(\tilde{\alpha})=\left\{\sum_{j=1}^{n}\alpha_{ij}-1,\sum_{i=1}^{n}\alpha_{ij}-1,i,j=\overline{1,n},u_i-u_j+n\alpha_{ij}-(n-1),i,j=\overline{2,n}\right\}\leqslant$$

$$\leqslant\left\{\sum_{j=1}^{n}\beta_{ij}-1,\sum_{i=1}^{n}\beta_{ij}-1,i,j=\overline{1,n},u_i-u_j+n\beta_{ij}-(n-1),i,j=\overline{2,n}\right\}=S(\tilde{\beta}).$$

Неравенства выполняются покомпонентно в силу $\alpha_{ij}\leqslant\beta_{ij}$, $\forall i,j=\overline{1,n}$. По условию леммы $(S(\tilde{\alpha})\leqslant 0)\Leftrightarrow(F_j(\tilde{\alpha})=0)$, $(S(\tilde{\beta})\leqslant 0)\Leftrightarrow(F_j(\tilde{\beta})=0)$. Учитывая, что $S(\tilde{\alpha})\preccurlyeq S(\tilde{\beta})$, имеем $(F_j(\tilde{\alpha})=1)\Rightarrow(F_j(\tilde{\beta})=1)$, поэтому $F_j(\tilde{\alpha})\leqslant F_j(\tilde{\beta})$. Следовательно, $F_j(\tilde{x})$ — монотонная *ФАЛ*.

Функция $F_0(\tilde{x})=\bigvee_{j=1}^{M}F_j(\tilde{x})$ является монотонной. Это следует из того, что класс монотонных *ФАЛ* является замкнутым и содержит дизъюнкцию. Функция $F_0(\tilde{x})$ равна нулю тогда и только тогда, когда выполняются все ограничения (2.14)–(2.16), так как дизъюнкция $\bigvee_{j=1}^{M}F_j(\tilde{x})$ равна нулю только при $F_j(\tilde{x})=0$ для всех $j=\overline{1,M}$. □

Следствие 2.2. *Утверждения леммы остаются справедливыми для моделей* (1.6)–(1.26) *многих коммивояжеров.*

Доказательство аналогично доказательству леммы 2.1.

Получили, что если область допустимых решений $\Omega=\{\tilde{x}\in B^N : F_0(\tilde{x})=0\}\neq\varnothing$, то решением задачи является верхний ноль функций $F_0(\tilde{x})$. Задача (2.17) сводится к задаче расшифровки монотонной *ФАЛ* или к поиску ее верхних нулей [50].

Псевдобулевая задача линейного программирования (2.17)

$$\min_{x\in\Omega}(c,\tilde{x}),\ \Omega=\left\{\tilde{x}\in B^N :\ F_0(\tilde{x})=0\right\} \tag{2.18}$$

с *ДНФ* ограничениями позволяет учитывать знания о решении *TSP*, которые представимы в *ДНФ* форме. Например, если необходимо включить прохождение дуг x_{kl} и x_{pm}, тогда к ограничениям добавляется условие $x_{kl}\&x_{pm} = 1$. Если, наоборот, не включать, то условие можно записать $x_{kl} \vee x_{pm} = 0$ [31].

Полученный формализм позволяет учитывать знания о модели и решениях, а также использовать их в теоретических обоснованиях и конкретных алгоритмах решения.

2.1.3 Модели псевдобулевой условной оптимизации с дизъюнктивными ограничениями для многоагентной задачи коммивояжера

Задаче $mTSP$ с общими интересами соответствует одна целевая функция, выражающая минимум общего расстояния, как и в задаче для одного коммивояжера. Ограничения линейные. Задачу можно привести к задаче псевдобулевой оптимизации с дизъюнктивными ограничениями.

При большой размерности задачи (сложность сети) условной псевдобулевой оптимизации с *ДНФ* ограничениями применять полиномиальные алгоритмы, предназначенные для такого класса задач может быть нерационально. Требуется упрощение задачи (применение приближенных, эвристических методов) с помощью отсечения излишних вариантов перебора на основе имеющихся знаний. Прежде всего снижение сложности (размерности) достигается с помощью кластеризации сети. При этом количество агентов, количество депо и их расположение, количество кластеров может быть задано, искомо или быть произвольным.

Так как любую задачу псевдобулевой оптимизации можно представить в эквивалентной форме с *ДНФ* ограничением, то это справедливо и для задач с линейной целевой функцией. Следовательно, решение любой линейной задачи (в том числе $mTSP$ после кластеризации) можно осуществлять по схеме, представленной на рис. 2.1.

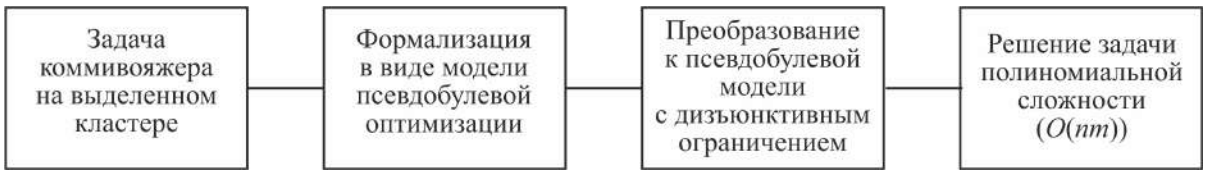


Рисунок 2.1 — Схема решения задачи $mTSP$

Следуя идеологии сведения исходной задачи к нескольким задачам меньшей размерности, предложен следующий подход к решению задачи маршрутизации для многих агентов. Пусть $m = 2$ (два агента). Приведем обобщенный алгоритм $AmTSP$ (алгоритм 2.1). Каждый шаг алгоритма $AmTSP$ конкретизируется в зависимости от структуры (сложности) исходной сети и всей имеющейся информации (знаний).

Алгоритм 2.1. *Решение задачи для двух коммивояжеров* ($AmTSP$)

Вход: *сеть* $S = (G, C)$, $G = (U, V)$, $n = |V|$, U — *множество дуг*, C — *матрица расстояний* (*весов*); *информация о структуре сети.*

Выход: *маршруты коммивояжеров, длина общего маршрута.*

1: *Провести кластеризацию сети*: $S = S_1 \cup S_2$, $G = G_1 \cup G_2$, $V = V_1 \cup V_2$ ($V_1 \cap V_2 = \varnothing$).

2: *На сетях* S_1, S_2 *сформировать TSP, выписать все основные и дополнительные ограничения.*

3: *Трансформировать TSP к задачам псевдобулевой оптимизации с ДНФ ограничениями.*

4: *Найти решения задач с ДНФ ограничениями.*

5: *Провести локальные преобразования, обмениваясь вершинами множеств* V_1, V_2. *Добавление вершины приводит к изменению ДНФ ограничений* (*добавление интервалов конъюнкций*). *Алгоритм остается полиномиальным.*

6: *Выбрать лучший вариант* (*или провести заданное число итераций*).

7: *Получить решение исходной задачи.*

Как было показано (лемма 2.1) задача каждого коммивояжера на выделенном кластере является задачей скалярной псевдобулевой условной оптимизации, т. е. может быть представлена в канонической форме с дизъюнктивными ограничениями (2.9):

$$f_k(\tilde{x}) = (c^k, \tilde{x}) = \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} c_{ij}^k x_{ij} \to min, \quad \bigvee_{j=1}^{m_k} x_{j_1}^{\sigma_{j_1}} \& \ldots \& x_{j_{r_j}}^{\sigma_{j_{r_j}}} = 1, \quad k = \overline{1,m}. \tag{2.19}$$

Каноническая модель является исчерпывающей в своем классе в силу полноты. Левая часть ограничения (2.19) является *ДНФ* характеристической функции множества Ω^k — ограничений искомой задачи на k-ом кластере, в которой уже учтена дополнительная информация о структуре кластера и искомого решения (запреты, предписания и др.). Рассмотрим алгоритм решения таких задач.

В случае общих интересов модель будет однокритериальной:

$$f_0(\tilde{x}) = \sum_{k=1}^{m} f_k(\tilde{x}) \to min, \quad \bigvee_{j=1}^{M} x_{j_1}^{\sigma_{j_1}} \& \ldots \& x_{j_{r_j}}^{\sigma_{j_{r_j}}} = 1. \tag{2.20}$$

Процессом выбора решения будем называть поиск такого набора значений $\alpha \in X^N$, $X \in \{0,1\}$ признаковых предикатов, чтобы (одновременно или по отдельности):

— обращался в единицу один или несколько целевых предикатов;

— достигала экстремального значения несколько (или одна в однокритериальной постановке) псевдобулевых функций $f_k, k = \overline{1,m}$.

Единственное ограничение канонической модели (2.20) в виде *ДНФ* характеристического множества ограничений задает И/ИЛИ граф, которому соответствует логическая система продукций. Существование логической системы продукций (*ЛСП*)

$$\begin{cases} x_{j_1}^{\sigma_{j_1}} \ldots x_{j_{r_j}}^{\sigma_{j_{r_j}}} \to g_j, \\ g_j \to g_0,\ j = \overline{1,M} \end{cases}$$

позволяет выводить целевой факт $g_0 =$«$\tilde{x}$ — допустимое решение». Граф И/ИЛИ ограничения канонической модели задачи $mTSP$ является трехъярусным.

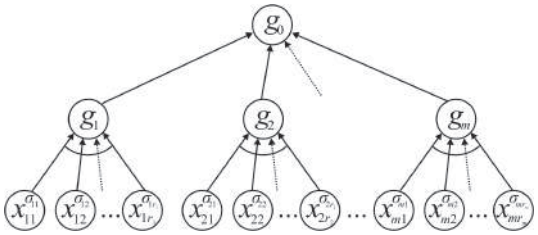


Рисунок 2.2 — И/ИЛИ граф *ДНФ* ограничения канонической модели

Любой граф *ЛСП*, не имеющий циклов, может быть сведен к трехъярусному и представлен в виде *ДНФ*. Откуда следует, что соответствующая база знаний (*БЗ*) системы построения допустимого решения $mTSP$ должна удовлетворять следующим требованиям:

1) решения $mTSP$ должны удовлетворять ограничениям задачи, следовательно, *ЛСП* должна обеспечивать возможность вывода целевых предикатов, соответствующих этим ограничениям;

2) группа ограничений, которые должны выполняться одновременно, задаются вершиной типа «И», связывающей эти ограничения вместе.

В работах Т. М. Леденевой и соавтров [12; 84—86] рассматривается нечеткий вывод на *ЛСП*.

Заметим, что *ДНФ* ограничение может быть получено с помощью обучения по прецедентной (эмпирической) информации [70; 173]. Для синтеза *ДНФ* по заданным *ЛСП* можно использовать *D*- и *DS*-алгоритмы [52], реализующие соответственно стратегии «сверху вниз» и «снизу вверх». То есть реализуется *синтез областей допустимости решений в знание-ориентированных продукционных системах моделей псевдобулевой условной оптимизации, соответствующих* $mTSP$ (в этих же работах можно найти оценки сложности алгоритмов).

Вопрос полноты знаний об ограничениях в *БЗ TSP* является важным и рассматривается самостоятельно в теории знаниеориентированных систем.

Неопределенность в кластерном представлении $mTSP$. Уточним шаги 4, 5 алгоритма $AmTSP$ с точки зрения преодоления неопределенности (для $m = 2$, аналогичная ситуация для $m > 2$). Задачи псевдобулевой оптимизации для каждого кластера имеют вид:

$$\begin{gathered} f_1(\tilde{x}) = (c^1, \tilde{x}) \to \min, \quad F_1(\tilde{x}) = 0, \\ f_2(\tilde{x}) = (c^2, \tilde{x}) \to \min, \quad F_2(\tilde{x}) = 0, \\ c^k = (c^k_{11}, c^k_{12}, \ldots, c^k_{nn}) \equiv (c^k_1, c^k_2, \ldots, c^k_j, \ldots, c^k_N), \\ \tilde{x} \in B^N, \quad N = n^2, \quad F_k(\tilde{x}) \in P_2(N), \quad k = 1{,}2. \end{gathered} \tag{2.21}$$

В том случае, когда кластеры определены единственным образом, а целевые функции и ограничения заданы точно, решение полученных задач на кластерах сводится к расшифровке монотонной функции алгебры логики (или к поиску ее верхних нулей). В более общем случае модели $mTSP$ получены как неполное представление исходной задачи $mTSP$, то есть когда в результате кластеризации (или другом сведении и задаче меньшей размерности) при

исследовании линейной модели не удалось получить полную информацию о ее ограничениях. Но, по доказанному выше, $F_j(\tilde{x})$ являются монотонными функциями алгебры логики.

Будем предполагать, что существуют множества

$$\begin{gathered} M_{k0}^{F_k} = \left\{\tilde{x} \in B^N : F_k(\tilde{x}) = 0\right\}; \quad M_{k1}^{F_k} = \left\{\tilde{x} \in B^N : F_k(\tilde{x}) = 1\right\}; \\ M_{k0}^{f_k} = \left\{\tilde{x} \in B^N : f_k(\tilde{x}) = 0\right\}; \quad M_{k1}^{f_k} = \left\{\tilde{x} \in B^N : f_k(\tilde{x}) = 1\right\}, \quad k = 1,2. \end{gathered} \tag{2.22}$$

С позиции первого коммивояжера ($k = 1$) $F_k(\tilde{x})$ — функция, определяющая допустимые решения, задана частично с помощью указания множеств наборов $M_{k0}^{f_k}, M_{k1}^{f_k}$ (прецедентов или фактов), т. е. заданы некоторые частичные функции алгебры логики $f_k, k = 1,2$.

Пусть Φ_k — множество монотонных функций алгебры логики из $P_2(n)$, принимающих значение «0» на множестве $M_{0k}^{f_k}$ и значение «1» на множестве $M_{1k}^{f_k}$, а $Z_k(\Phi_k)$ — множество всех верхних нулей всех функций из Φ_k.

Непротиворечивым решением задач (2.21) называется такой набор $\tilde{z}_k^* \in Z_k(\Phi_k)$, что

$$\sum_{j=1}^{N} c_j^k z_j^* = \min_{z \in Z_k(\Phi_k)} \sum_{j=1}^{N} c_j^k z_j.$$

Не теряя общности, можно считать, что булевы переменные упорядочены так, что $c_1^k > \ldots > c_N^k$. Это легко выполнить для любой исходной задачи.

Теорема 2.4. *Функция $f_k \in P_2(N)$, не являющаяся константой, монотонна тогда и только тогда, когда для любых пар вершин $\tilde{x},\tilde{y} \in B^N$ таких, что $f_k(\tilde{x}) = 1$, $f_k(\tilde{y}) = 0$, найдется переменная с номером $i \in \{1,2,\ldots,N\}$ такая, что $x_i = 1$, $y_i = 0$.*

Доказательство. Необходимость. Докажем необходимость методом от противного. Пусть $f_k \in P_2(N)$ не константа, монотонна и не существует переменной с номером $i \in \{1,2,\ldots,N\}$ такой, что $x_i = 1$, $y_i = 0$, т. е. $x_i \leqslant y_i$, $i = \overline{1,N}$. Тогда $\tilde{x} \preccurlyeq \tilde{y}$, но $f_k(\tilde{x}) > f_k(\tilde{y})$, что противоречит условию монотонности функции f_k.

Достаточность. Рассмотрим три множества пар наборов $\tilde{x},\tilde{y} \in B^N$:

$$W_{k1} = \{(\tilde{x},\tilde{y}) : \ f_k(\tilde{x}) = 1, \ f_k(\tilde{y}) = 0\};$$

$$W_{k2} = \{(\tilde{x},\tilde{y}) : \ f_k(\tilde{x}) = 0, \ f_k(\tilde{y}) = 1\};$$

$$W_{k3} = \{(\tilde{x},\tilde{y}) : \ f_k(\tilde{x}) = f_k(\tilde{y})\}.$$

Пусть $\tilde{x} \in W_{k1}$. Тогда $f_k(\tilde{x}) > f_k(\tilde{y})$ и по условию теоремы $\exists i : \ x_i > y_i$. Следовательно, либо $\tilde{x} \succ \tilde{y}$, либо наборы $\tilde{x}$ и $\tilde{y}$ — несравнимы. Для всех сравнимых наборов из W_{k1} имеем: $\tilde{x} \succ \tilde{y}$ и $f_k(\tilde{x}) > f_k(\tilde{y})$.

Аналогично проверяется выполнение условия монотонности функции f_k на множества W_{k2}.

Пусть $\tilde{x} \in W_{k3}$, тогда $f_k(\tilde{x}) = f_k(\tilde{y})$, в том числе всегда, когда $\tilde{x} \preccurlyeq \tilde{y}$. Учитывая, что объединение $W_{k1} \cup W_{k2} \cup W_{k3}$ содержит любую пару вершин куба B^N, получаем, что f_k — монотонная функция: если $\tilde{x} \preccurlyeq \tilde{y}$, то $f_k(\tilde{x}) \leqslant f_k(\tilde{y})$. □

На основании теоремы 2.4 можно сделать следующий вывод. Если во множествах $M_{k0}^{f_k}$ и $M_{k1}^{f_k}$ частичной функции алгебры логики f_k найдутся такие наборы $\tilde{\alpha} \in M_{k0}^{f_k}$ и $\tilde{\beta} \in M_{k1}^{f_k}$, что не существует переменной с номером $i \in \{1,\ldots,N\}$, для которой $\alpha_i < \beta_i$, то f_k не может быть доопределена монотонной функцией.

Пусть частичная функция f_k доопределена монотонной функцией φ_k. Необходимо, чтобы $M_{k1}^{f_k} \subseteq M_{k1}^{\varphi_k}$, $M_{k0}^{f_k} \subseteq M_{k0}^{\varphi_k}$, следовательно, любой набор из $M_{k1}^{f_k}$ должен покрываться некоторым интервалом $N_j^k \subseteq M_{k1}^{\varphi_k}$, но N_j^k не должен содержать точек из $M_{k0}^{f_k}$, каждый набор из $M_{k0}^{f_k}$ должен покрываться некоторым интервалом $N_L^k \subseteq M_{k0}^{\varphi_k}$, но N_L^k не должен содержать точек из $M_{k1}^{f_k}$. Класс монотонных функций φ_k, доопределяющих f_k, обозначим $\Phi_k \subset M_k$. Функции класса Φ_k определяют множество:

$$\overline{\Phi}_k = \{g_k \in P_2(N) :\ g_k(\tilde{x}) = \overline{\varphi}_k(\tilde{x}), \varphi_k \in \Phi_k\}.$$

Любая функция может быть представлена сокращенной *ДНФ* так, что может быть указан набор максимальных вне $M_{k1}^{f_k}$ интервалов, покрывающих все точки из множества $M_{k0}^{f_k}$.

Множества $M_{k0}^{f_k}$ и $M_{k1}^{f_k}$, являющиеся частью исходной информации в задаче (2.21) и содержащие $(m_0^k + m_1^k)$ двоичных наборов, можно рассматривать, как стандартную обучающую информацию задачи Z_k распознавания: в обучающей таблице $T_{m_0^k m_1^k}^k = M_{k0}^{f_k} \cup M_{k1}^{f_k}$, наборы $\tilde{x} \in M_{k0}^{f_k}$ относятся к классу K_1^k допустимых решений задачи (2.21), а $\tilde{x} \in M_{k1}^{f_k}$ — к классу K_2^k недопустимых решений.

Рассмотрим любой максимальный интервал $N_L^k \subseteq M_{k1}^{g_1}$ произвольной функции $g_k \in \overline{\Phi}_k$ и соответствующую ему элементарную конъюнкцию $L = \overline{x}_{i_1} \& \ldots \& \overline{x}_{i_r}$. Вхождение переменных в простую импликанту только с инверсиями доказывается с учетом монотонности функции $\overline{g}_k(\tilde{x})$.

Любой набор $\tilde{\alpha} \in N_L^k$ является допустимым решением и в этом наборе $\alpha_{i_1} = 0, \ldots, \alpha_{i_r} = 0$. Среди всех наборов $\tilde{\alpha} \in N_L^k$ наибольшее значение целевой функции будет достигаться на наборе, в котором $\alpha_j = 1$ для всех j из множества $\{1,2,\ldots,N\} \setminus \{i_1,i_2,\ldots,i_r\}$. Назовем такой набор экстремальным.

Если теперь для каждой простой импликанты всех функций из множества Φ_k выбрать экстремальный набор, то в полученном множестве будут содержаться все непротиворечивые решения задачи.

Различные доопределения функции f_k функциями $\varphi_k \in \Phi_k$ отличаются значениями $\varphi_k(\tilde{x})$ на множестве $B^N \setminus \left\{ M_{k0}^{f_k} \cup M_{k1}^{f_k} \right\}$, поэтому простые импликанты различных функций g_k из $\overline{\Phi}_k$ могут отличаться рангом. Экстремальная постановка задачи требует из всех простых импликант всех функций $g_k \in \overline{\Phi}_k$ выделить кратчайшие. Для построения таких простых импликант с инверсиями, необходимыми для любых доопределений, можно использовать следующий

Алгоритм 2.2. *Построение простых импликант*

1: *Для каждого набора* $\tilde{\alpha} \in M_{k0}^{f_k}$ *выписать конъюнктивную нормальную форму (КНФ)* $K_k(\tilde{\alpha})$*, каждая дизъюнкция которой состоит из переменных* $\overline{x}_i$ *(с инверсиями) таких, что* $\alpha_i < \beta_i$ *для одного из наборов* $\tilde{\beta} \in M_{k1}^{f_k}$*; КНФ* $K_k(\tilde{\alpha})$ *будет содержать* $m_1^k = \left|M_{k1}^{f_k}\right|$ *дизъюнкций — число наборов в множестве* $M_{k1}^{f_k}$*.*

2: *В полученных КНФ* $K_k(\tilde{\alpha}_1), \ldots, K_k(\tilde{\alpha}_{m_0^k})$*, где* $m_0^k = \left|M_{k0}^{f_k}\right|$ *(число наборов в* $M_{k0}^{f_k}$*) раскрыть скобки и выполнить операции поглощения, получая ДНФ* $D_1, \ldots, D_{m_0^k}$*.*

3: *Записать ДНФ* $D_1 \vee \ldots \vee D_{m_0}$ *и выполнить все возможные операции поглощения. Будет получена ДНФ* $D(\overline{\Phi}_k)$*.*

Обозначим через $A_z^k = A_z^k(T_{m_0^k m_1^k}, \tilde{x})$ алгоритм распознавания класса произвольного набора $\tilde{x} \in B^N \backslash T_{m_0^k m_1^k}$; пусть A_Z^{k*} — корректный алгоритм:

$$A_z^{k*}(T_{m_0^k m_1^k}, \tilde{x}) = \begin{cases} 1,\ \tilde{x} \in K_2^k = B^n \backslash \Omega_k \\ 0,\ \tilde{x} \in K_1^k = \Omega_k,\ k = 1{,}2 \end{cases}$$

Очевидно, что если информация в $T_{m_0^k m_1^k}$ достоверна, и алгоритм A_z^{k*} относит экстремальный набор $\tilde{x}^*$, являющийся непротиворечивым решением задачи (2.21), к классу $K_1^k = \Omega_k$, то $\tilde{x}^*$ является решением задачи

$$\min \sum_{i=1}^{n} c_i^k x_i / \tilde{x} \in \Omega_k.$$

Пусть алгоритм A_z^k — экстремальный в некотором классе алгоритмов распознавания или построен с применением корректирующих (алгебраических) методов, т. е. является в некотором смысле наилучшим для решения задачи $Z_k(T_{m_0^k m_1^k}, \tilde{x})$.

Подход к решению $mTSP$ как задачи линейного псевдобулевого программирования с частично заданными ограничениями с применением алгоритмов распознавания образов состоит в следующем:

1. При помощи алгоритма находится множество экстремальных наборов $\aleph^k = \{\tilde{x}^*\}$ для задачи (2.21)-(2.22).
2. Алгоритм A_z^k определяет принадлежность экстремальных наборов из $\aleph^k$ к классу K_1; $\aleph_A^k \subseteq \aleph^k$; $\aleph_A^k = \left\{\tilde{x}^* \in \aleph^k : A(T_{m_0^k m_1^k}, \tilde{x}^*) = 0\right\}$.
3. Если $\aleph_A^k \neq 0$, то входящий в него экстремальный набор, которому соответствует наибольшее значение целевой функции, объявляется решением задачи.
4. Если $\aleph_A^k = 0$, то к множеству $M_{k1}^{f_k}$ добавляются наборы $\aleph^k$, т. е. $M_{k1}^{f_k} := M_{k1}^{f_k} \cup \aleph^k$, и повторяется пункт 1, внутри которого обеспечивается проверка монотонности, обеспечивающая линейность модели.

Замечание 2.1. *Добавление к множеству* $M_{k1}^{f_k}$ *множества экстремальных наборов* $\aleph^k$ *равносильно переопределению для некоторых функций алгебры логики верхних нулей единицами.*

Линейность задачи $mTSP$ позволила эффективно «сузить» область поиска решения, что обеспечивается указанным алгоритмом (см. [72] по сужающим запросам).

2.2 Многокритериальные задачи многих коммивояжеров, представленные в канонической форме

Пусть задача $mTSP$ сводится к многокритериальной псевдобулевой оптимизации с дизъюнктивным ограничением:

$$\begin{cases} \min f_1(\tilde{x}),\ \min f_2(\tilde{x}), \ldots, \min f_m(\tilde{x}), \\ \bigvee\limits_{j=1}^{m} x_{j_1}^{\sigma_{j_1}} \& \ldots \& x_{j_{r_j}}^{\sigma_{j_{r_j}}} = 1, \\ f_k \in LPS_2(N), \quad k = \overline{1,m}, \quad \tilde{x} \in B^N. \end{cases} \tag{2.23}$$

Подчеркнем, что задачи псевдобулевой оптимизации возникают как результат синтеза моделей $mTSP$ на основе индуктивного обобщения или построения логического описания области дедуктивной выводимости в системах, основанных на знаниях.

Необходимо найти паретовское множество $\mathcal{P}$ задачи (2.23), его логическое описание в виде *ДНФ* и подходов к выбору решения $\tilde{x}^* \in \mathcal{P}$. Для этого используем необходимое условие принадлежности точки множеству Парето и принцип ветвей и границ.

В задаче $mTSP$ учитывается информация о распределении весов дуг. Выбор прохождения тех или иных дуг для коммивояжера зависит от среднего значения веса дуги, дисперсии (при большой дисперсии преобладают дуги с большими весами). Если в матрице весов вычесть среднее значение веса, получим новую матрицу весов с положительными и отрицательными значениями. Такие матрицы появляются в процессе реализации некоторых алгоритмов TSP. Поэтому необходимое условие принадлежности точки множеству Парето учитывает знаки коэффициентов c_j^k, $j = \overline{1,N}$, $N = n^2$, $k = \overline{1,m}$.

Рассмотрим необходимое условие принадлежности точки множеству Парето в задаче безусловной оптимизации.

Обозначим P_i множество номеров переменных, имеющих положительный, а N_i — множество номеров переменных, имеющих отрицательный коэффициент в линейной функции f_i, $i = \overline{1,m}$. Пусть

$$P_0 = P_1 \cap P_2 \cap \ldots \cap P_m, \quad N_0 = N_1 \cap N_2 \cap \ldots \cap N_m.$$

Лемма 2.2. *Если для задачи безусловной многокритериальной оптимизации $mTSP$*

$$\begin{cases} \min f_1(\tilde{x}),\ \min f_2(\tilde{x}), \ldots,\ \min f_m(\tilde{x}) \\ \tilde{x} \in B^N,\ f_1, \ldots, f_m \in LPS_2(N) \end{cases} \tag{2.24}$$

множества P_0 и N_0 непусты и точка $\tilde{x}^*$ является паретовской, то она удовлетворяет уравнению

$$\left(\mathop{\&}_{i \in P_0} x_i \right) \left(\mathop{\&}_{i \in N_0} \overline{x}_i \right) = 1. \tag{2.25}$$

Доказательство. Пусть $\tilde{\alpha}$ — любая точка, удовлетворяющая уравнению (2.25). Тогда найдется такое i, что $\alpha_i = 0$ при $i \in P_1 \cap P_2 \cap \ldots \cap P_m$ или $\alpha_i = 1$ при $i \in N_1 \cap N_2 \cap \ldots \cap N_m$. Заменяя α_i на $\overline{\alpha}_i$, получим точку $\tilde{\alpha}'$ такую, что $f_1(\tilde{\alpha}') < f_1(\tilde{\alpha})$, $f_2(\tilde{\alpha}') < f_2(\tilde{\alpha}), \ldots, f_m(\tilde{\alpha}') < f_m(\tilde{\alpha})$ и тогда $\tilde{\alpha}$ — не является паретовской точкой. □

Замечание 2.2. *Если* $P_0 \neq \varnothing$ *и* $N_0 \neq \varnothing$, *то необходимыми условиями эффективности точки* $\tilde{x}^*$ *в задаче* (2.24) *являются* $\underset{i\in P_0}{\&} x_i = 1$ *и* $\underset{i\in N_0}{\&} \overline{x}_i = 1$. *Если* $P_0 \neq \varnothing$ *и* $N_0 = \varnothing$, *то необходимым условием эффективности точки* $\tilde{x}^*$ *в задаче* (2.24) *является* $\underset{i\in P_0}{\&} x_i = 1$. *Если* $P_0 = \varnothing$ *и* $N_0 \neq \varnothing$, *то необходимым условием эффективности точки* $\tilde{x}^*$ *в задаче* (2.24) *является* $\underset{i\in N_0}{\&} \overline{x}_i = 1$.

Определение 2.6. *Нижней векторной оценкой допустимого множества* X *называется вектор* $\left(\min\limits_{\tilde{x}\in X} f_1(\tilde{x}), \min\limits_{\tilde{x}\in X} f_2(\tilde{x}), \ldots, \min\limits_{\tilde{x}\in X} f_m(\tilde{x})\right)$.

Определение 2.7. *Вектор* $(a_1, a_2, \ldots, a_m)$ *мажорируется вектором* $(b_1, b_2, \ldots, b_m)$, *если* $a_j \leqslant b_j$ *для всех* $j = \overline{1,m}$, *причем хотя бы для одного* j *выполняется строго неравенство* $a_j < b_j$.

Определение 2.8. *Рекордом называется вектор значений скалярных критериев в некоторой допустимой точке* $\tilde{\gamma}$, *который не мажорируется никаким другим имеющимся рекордом или нижней векторной оценкой, полученной для какого-либо подмножества допустимого множества решений.*

Будем использовать метод ветвей и границ (см. [52; 71] для данного класса задач). Ветвление будем осуществлять путем фиксации значений 0 и 1 переменных x_i, $i = \overline{1,N}$. На каждом шаге ветвления будет происходить измельчение множества B^N и порождение подмножеств-интервалов, подлежащих исследованию.

Интервал подлежит исключению из рассмотрения в следующих случаях:

1. Существует рекорд, мажорирующий верхнюю векторную оценку этого интервала.
2. Известен другой интервал, нижняя векторная оценка которого мажорирует верхнюю векторную оценку этого интервала.

Интервал подлежит ветвлению, если он не подлежит исключению и его верхняя векторная оценка отличается от нижней. Выбор переменной и интервала, подлежащего ветвлению, является эвристическим элементом метода и будет рассмотрен далее.

Рассмотрим задачу $mTSP$ (2.24) с добавлением дизъюнктивных ограничений.

Теорема 2.5. *Пусть в задаче* $mTSP$ (2.24) *существует непустое множество Парето* $\mathcal{P}$, *и к данной задаче добавляется ограничение* $\tilde{x} \in \Omega;\ \Omega \neq \varnothing;\ \Omega \subset B^N,\ \Omega \neq B^N$. *Для полученной задачи множество* $\mathcal{P} \cap \Omega$, *если оно не пусто, будет состоять только из паретовских точек.*

Доказательство. Пусть $\tilde{x}^* \in \{\mathcal{P} \cap \Omega\}$. Тогда $\tilde{x}^* \in \mathcal{P}$ и не мажорируется ни одной точкой из B^N и, тем более, — ни одной точкой из $\mathcal{P} \cap \Omega$, а так как $\tilde{x}^* \in \Omega$, то она является допустимой. Таким образом, $\tilde{x}^*$ есть немажорируемая допустимая точка, следовательно, является паретовской. □

Условие (2.25) не является достаточным. В этом можно убедиться, рассмотрев следующий упрощенный пример:

$$\begin{cases} \min f_1(\tilde{x}) = -25x_1 - x_2 + x_3 + x_4; \\ \min f_2(\tilde{x}) = x_1 - x_2 + x_3 - 25x_4; \\ \tilde{x} \in B^N. \end{cases}$$

Очевидно, что $P_1 \cap P_2 = \{2\}$; $N_1 \cap N_2 = \{3\}$. Условие (2.25) принимает вид: $x_2\overline{x}_3 = 1$. Этому условию удовлетворяет точка $\tilde{\beta} = (0,1,0,0)$, но она не паретовская: взяв точку $\tilde{\gamma} = (1,1,0,1)$, убеждаемся, что $-25 = f_1(\tilde{\gamma}) < f_1(\tilde{\beta}) = -1$, $-25 = f_2(\tilde{\gamma}) < f_2(\tilde{\beta}) = -1$.

Найдем множество Парето в задаче безусловной многокритериальной оптимизации. Используем необходимое условие, которому должны удовлетворять эффективные точки: $x_2\overline{x}_3 = 1$. Начальное дерево имеет вид:

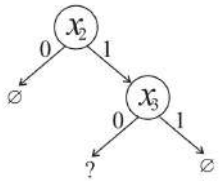


Рисунок 2.3 — Начальное дерево

Знак «∅» указывает на отсутствие эффективных точек в интервале соответствующей ветви; знак «?» — на необходимость дальнейшего ветвления; «×» — на отсечение интервала.

Обозначим $H(y_1, y_2)$ — вектор верхних и $L(y_1, y_2)$ — вектор нижних достижимых оценок функций f_1, f_2, $R(\alpha,\beta)$ — рекорд.

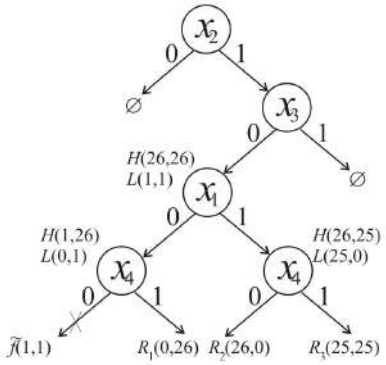


Рисунок 2.4 — Результат решения задачи

Множество Парето состоит из трех точек $\{0101, 1100, 1101\}$, являющихся рекордными (R_1, R_2, R_3) и имеет логическое описание: $\mathcal{P} = \{\tilde{x} : \ x_1x_2\overline{x}_3 \bigvee x_2\overline{x}_3x_4 = 1\}$.

Замечание 2.3. *Условие $\tilde{x} \in \mathcal{P} \cap \Omega$, как следует из теоремы* 2.5*, является достаточным для того, чтобы точка $\tilde{x}$ была паретовской в задаче с ограничением $\tilde{x} \in \Omega$ при $\mathcal{P} \cap \Omega \neq \varnothing$. Однако это условие не является необходимым. Действительно, в задаче*

$$\begin{cases} \min f_1(\tilde{x}) = -25x_1 - x_2 + x_3 + x_4; \\ \min f_2(\tilde{x}) = x_1 - x_2 + x_3 - 25x_4; \\ \tilde{x} \in \Omega = \{\tilde{x} : \ \overline{x}_1x_2 = 1\} \subset B^N \end{cases}$$

паретовскими являются точки $\{0100, 0101\}$ *со значениями векторов критериев* $(-1, -1)$ *и* $(0, -26)$ *соответственно, причем точка* $\{0100\}$ *не принадлежит множеству* $\mathcal{P} \cap \Omega$.

Рассмотрим варианты выбора интервалов и переменных для ветвления.

От последовательности выбора интервалов и переменных для ветвления зависит скорость нахождения решения задачи. Стратегии ветвления являются эвристиками, например (возможны другие):

1. Разбиению по переменной с номером i подвергается тот интервал множества допустимых решений, конъюнкция которого не содержит литерала переменной с номером i, и изменение этой переменной с единицы на нуль обеспечивает одновременное уменьшение как можно большего числа скалярных критериев.
2. Разбиению подвергается тот интервал, для которого является максимальной следующая мера различия между верхней $H = (h_1, \ldots, h_m)$ и нижней $L = (l_1, \ldots, l_m)$ его векторными оценками:
$$D(H,L) = \min_{1 \leqslant j \leqslant m} \left(\frac{h_j - l_j}{M_j - \mu_j} \right),$$
где $M_j = \max\limits_{\tilde{x} \in \Omega} f_j(\tilde{x})$; $\mu_j = \min\limits_{\tilde{x} \in \Omega} f_j(\tilde{x})$; $M_j - \mu_j > 0$, поскольку в противном случае критерий f_j может быть исключен из рассмотрения.

Можно привести пример задачи многокритериальной псевдобулевой оптимизации $mTSP$, для которой процесс принятия решения по изложенному методу будет близок к полному перебору. В расчете на такие ситуации возможен приближенный подход к решению, суть которого состоит в следующем.

Пусть заданы значения $\varepsilon_1, \ldots, \varepsilon_m \in \mathbb{R}^+$. Будем говорить, что два вектора $(a_1, a_2, \ldots, a_m)$ и $(b_1, b_2, \ldots, b_m)$ являются $\tilde{\varepsilon}$-равными, если $|a_i - b_i| \leqslant \varepsilon_i$, $i = \overline{1,m}$.

Если верхняя и нижняя векторные оценки некоторого интервала, полученного при ветвлении, $\tilde{\varepsilon}$-равны, то такой интервал называется $\tilde{\varepsilon}$-интервалом.

Если нижняя векторная оценка некоторого $\tilde{\varepsilon}$-интервала мажорируется рекордом или нижней векторной оценкой другого интервала, то такой $\tilde{\varepsilon}$-интервал подлежит исключению.

Совокупность немажорируемых $\tilde{\varepsilon}$-интервалов вместе с рекордами дает приближение к искомому паретовскому множеству. Логическое описание паретовского множества $\mathcal{P}$ получается обратным проходом по ветвям деревьев ветвлений, листья которых соответствуют немажорируемым элементам.

Таким образом, предложены методы решения многокритериальных задач $mTSP$, представленных в канонической форме. Показано, как использование метода ветвей и границ решения таких задач позволяет строить логическое описание паретовского множества.

2.3 Методы постоптимального анализа в многоагентных задачах

2.3.1 Устойчивость и реоптимизация оптимальных маршрутов

Устойчивость оптимальных маршрутов в *TSP* при изменении матрицы весов рассматривалась учеными, начиная с 1970 года. Значительные результаты для ряда общих постановок *ЗКО* при возмущении весовой функции получены в Вычислительном центре РАН под руководством проф. В. К. Леонтьева [39; 40; 88]; в Белорусском госуниверситете под руководством проф. В. А. Емеличева [53—55]; в Институте кибернетики НАН Украины под руководством академика И. В. Сергиенко [113—116]; в Польской АН под руководством М. Либуры [216; 217; 244; 248]. Работы, посвященные исследованию вопросов устойчивости алгоритмов решения задач линейного целочисленного программирования на основе анализа устойчивости лексико-графических структур, ведутся в Омском госуниверситете под руководством проф. А. А. Колоколова [44—46].

В разделе 1.1.1 рассмотрены полиномиально разрешимые TSP (*ЗДО*), показано, что имеются результаты, связанные с понятием шаблонного маршрута (master tour) и решениями *TSP* на основе кривых Пеано [153; 158; 159; 226]. Так, в [139] ставится задача поиска маршрута, сужение которого (без потери исходного порядка обхода) на всякое подмножество множества исходных вершин позволяет получить «достаточно близкий» к оптимальному «подмаршрут». Более глубокое исследование задачи поиска подобных «универсальных» маршрутов, получившей соответствующее название universal *TSP*, проведено в работах [198; 234; 240]. Для шаблонных маршрутов в [170] показано, что для *TSP* с симметричной матрицей весов существует перенумерация вершин, сводящая данную матрицу к виду матрицы Кальмансона [202] тогда и только тогда, когда оптимальное решение исходной задачи есть шаблонный маршрут.

Вышеуказанные работы, посвященные устойчивости оптимальных решений в *ЗДО*, связаны с изучением устойчивости при изменении некоторой функции стоимости при неизменном множестве исходных данных и с построением шаблонного маршрута в *TSP*, всякий подмаршрут которого является оптимальным. В общем случае исследование возможности конструктивного использования найденного оптимального решения при построении оптимального решения задачи с возмущенными начальными данными связано с понятием реоптимизации.

Особенный интерес реоптимизация представляет в NP-трудных задачах. Реоптимизации оптимальных маршрутов в *TSP* посвящены работы [151; 157; 162; 177; 230; 231]. В NP-трудных задачах (и в частности в *TSP*) даже при известном оптимальном решении его реоптимизация, как правило, оказывается NP-трудной [143; 227; 230]. Ситуация не меняется даже в случае, когда известны все оптимальные решения исходной задачи. Также существует понятие «онлайн-оптимизация», когда ставится задача динамического изменения решения

задачи в ходе постоянно и быстро изменяющихся исходных данных, с использованием предыдущих решений [152; 164; 165; 199; 237].

Приложения, связанные с решением *TSP* и задачи распределения заданий в атомной энергетике, экологии и таксономии животных, задачи, связанные с оптимизацией перемещений бригады исполнителей в агрессивной внешней среде, изучаются учеными под руководством А. Г. Ченцова [97; 117—119]).

«Устойчивость — одно из наиболее общих понятий в современной науке. В математике рассматривается устойчивость решения некоторой задачи при изменении начальных данных. При наличии такой устойчивости полученное решение будет верным для целого множества различных входных параметров задачи, что особенно важно при применении математических результатов в области физики и инженерных наук, где начальные данные априори допускают погрешность. Именно благодаря широкому спектру актуальных приложений математическая теория устойчивости стала важной и неотъемлемой частью непрерывной математики» [80].

Одним из новых направлений является исследование устойчивости решений дискретных задач, например *ЗКО* [102]. В этих задачах, как правило, нужно отыскать оптимальный в некотором смысле объект среди большого числа похожих объектов [235]. Число таких объектов часто факториально или экспоненциально зависит от длины записи исходных данных задачи и, следовательно, прямой перебор неэффективен или фактически невозможен (NP-трудные задачи [42]).

Для многих *ЗКО* существуют более эффективные методы решения в сравнении с полным перебором (метод ветвей и границ [213], как метод линейного программирования [65]; метод динамического программирования [8]). Однако все это не позволяет избежать высоких вычислительных затрат при решении важных с прикладной точки зрения *ЗКО*. Поэтому устойчивость является удобным инструментом, позволяющим качественно исследовать поведение задачи, избегая ее трудоемкого решения.

Одним из направлений исследования устойчивости *ЗДО* можно считать исследование «радиуса устойчивости» некоторых величин, в пределах которых изменение функции стоимости не приводит к смене оптимального решения. Например, для *TSP* в [216] исследуется «радиус», в пределах которого можно изменять веса ребер, не нарушая найденного оптимального порядка обхода вершин.

В [63]: «исследуются условия, обеспечивающие возможность простой в вычислительном отношении адаптации найденного оптимального решения к добавлению, удалению или замене одного из элементов начальных данных с сохранением оптимальности. Важной особенностью данных исследований является возможность анализа адаптивной устойчивости оптимального решения при росте числа элементов входных данных (и открывающаяся с этим возможность полиномиальных решений некоторых постановок NP-трудных задач)». Подход, предложенный в [63], основан на анализе внутренней структуры *ЗКО* с помощью близких к методу динамического программирования соотношений.

Исследование устойчивости оптимальных решений к возмущению начальных данных задачи является типичной проблемой постоптимального анализа. В. А. Емеличев выделя-

ет два подхода к исследованию устойчивости в дискретных задачах. В первом подходе [54] ставится задача поиска условий, при которых оптимальное решение устойчиво к «малым» возмущениям начальных данных: «... под устойчивостью задачи чаще всего понимается одно из свойств непрерывности или полунепрерывности... оптимального отображения, т. е. точечно-множественного (многозначного) отображения, которое каждому набору параметров задачи ставит в соответствие множество искомых решений...» [54]. Иными словами ставится задача поиска «... такой окрестности исходных данных в пространстве параметров задачи, что любая «возмущенная» задача с параметрами из этой окрестности обладает некоторым свойством инвариантности по отношению к исходной задаче. Например, полунепрерывность сверху (снизу) в смысле Хаусдорфа оптимального отображения в задачах дискретной оптимизации превращается в свойство непоявления новых (сохранения исходных) оптимальных решений задачи при любых изменениях ее параметров в пределах «малой» окрестности исходных данных...» [54].

Другой подход связывается с количественной характеристикой такой области устойчивости. Например, ищется радиус предельного шара устойчивости в пространстве параметров задачи, возмущения начальных данных внутри которого не приводят к появлению новых (обеспечивается сохранение исходных) оптимальных решений.

Концепция реоптимизации описываются в работах [143; 151; 157; 162; 167; 172; 177; 206; 227; 230; 231; 233; 243]. Здесь рассматриваются произвольные возмущения данных задачи, включая изменение множества исходных данных при добавлении или удалении новых элементов. При добавлении новых элементов исходных данных решения задачи не могут сохраняться в исходном виде. В таком случае нужны алгоритмы, позволяющие реоптимизировать (адаптировать) полученное оптимальное решение к изменившимся начальным данным.

Как указывалось выше, одним из подходов к сохранению оптимальных маршрутов в задаче коммивояжера при изменении множества посещаемых вершин является подход, связанный с исследованием шаблонных маршрутов (master tour) [170]. По определению «маршрут $\alpha = (a_1, \ldots, a_n, a_1)$ является шаблонным маршрутом на множестве $\{a_1, \ldots, a_n\}$, если для любого подмножества $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$: $i_1 < \ldots < i_k$ маршрут $(a_{i_1}, \ldots, a_{i_k}, a_{i_1})$ является оптимальным на множестве $\{a_{i_1}, \ldots, a_{i_k}\}$». В работе [170] показано, что «симметричная матрица весов TSP допускает существование шаблонного маршрута тогда и только тогда, когда для некоторой перенумерации вершин выполнены условия Кальмансона [202]: $d_{ij} + d_{kl} \leqslant d_{ik} + d_{jl}$ и $d_{il} + d_{jk} \leqslant d_{ik} + d_{jl}$ для всяких $1 \leqslant i < j < k < l \leqslant n$». Для задач $mTSP$ на сложных сетях выявление знаний о существовании шаблонных маршрутов является самостоятельной задачей.

В связи с принятой в работе методологией по снижению размерности задачи маршрутизации многих коммивояжеров, базирующейся на адаптивной кластеризации сети (например, с помощью процедур максимального разреза), в работе рассматривались возможности реоптимизации в задаче о максимальном разрезе ($MAX\text{-}CUT$). Сравнительный анализ алгоритмов $MAX\text{-}CUT$ изложен в главе 3. В зависимости от наличия знаний о структуре сети

применим широкий спектр методов и алгоритмов кластеризации и декомпозиции. В этом случае важными являются параметры разреженности сети и ее блочный характер.

Кластеризация сети, ориентированная на поиск маршрута в кластере существенно меньшего размера, чем исходная сеть, позволяет применять точные алгоритмы (динамического программирования, метода ветвей и границ) или алгоритмы приближенного решения, метаэвристики в случае больших размеров кластера.

Предположим, что кластеризация сети по числу агентов проведена. На каждом кластере решена *TSP*, и результаты для всей сети не являются оптимальными (или приемлемыми). Итерация улучшения решения достигается обменом вершин между кластерами. Реоптимизация на каждом кластере позволяет сократить число операций для всей задачи. Наследование решения (т. е. использование уже найденного решения для поиска нового решения в случае изменения структуры сети кластера, например, добавление или удаление вершины) является важной составляющей для разработки полиномиальных алгоритмов.

Реоптимизация затрагивает уровень межкластерного анализа, а также задачи на кластерах. При этом предполагается, что полученные кластеры имеют такую размерность (число вершин), что на каждом кластере можно применять точные методы решения.

Для наглядности рассмотрим следующий случай.

Задача. Пусть необходимо найти m маршрутов для m агентов-коммивояжеров на сети $S = (G, C)$, где $G = (V, U)$ — граф, у которого V — множество вершин, $|V| = n$; U — множество дуг (i,j), $i, j \in V$; C — разреженная асимметричная матрица весов (расстояний), $c_{ij} > 0$, $c_{ij} \neq c_{ji}$. Коммивояжеры не конкурируют друг с другом, а на длину каждого маршрута дополнительных условий не накладывается.

Решение задачи может быть найдено с помощью линейной задачи о назначениях (*ЛЗН*), в результате решения которой получаем k циклов. Если $k = m$, то задача решена. Если $k > m$ или $k < m$, то решение получаем в результате перебора вариантов разрыва (и слияния) некоторых циклов и образования m циклов.

Рассмотрим более подробно *TSP* на кластере $G(I,U), |I| = n$:

$$\begin{gathered} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to min, \\ \sum_{i=1}^{n} x_{ij} = 1,\ \sum_{j=1}^{n} x_{ij} = 1,\ x_{ij} \geqslant 0,\ i,j = \overline{1,n}, \\ u_i - v_j + nx_{ij} \leqslant n-1,\ i,j = \overline{2,n},\ i \neq j. \end{gathered} \tag{2.26}$$

Для асимметричной $c_{ij} \neq c_{ji}$ разреженной матрицы C эффективным является алгоритм точного метода ветвей и границ, для которого в качестве дерева вариантов используется решение *ЛЗН* [182; 221]. Если циклов более одного, то производится перебор вариантов разрыва циклов. В этом случае размерность дерева решений оказывается наименьшей. Рекурсия обхода дерева *ЛЗН* строится по матрице расстояний C, а разрывы циклов задаются приписыванием (запрещенной) дуге разрыва бесконечного значения длины дуги.

Основываясь на реоптимизации задачи о назначении [107], в которой используется разностная схема организации ветвления, получена оценка сложности алгоритма $O(n^2)$ в отличии от прямолинейной реализации метода ветвей и границ; а для асимметричной *TSP* — оценка сложности $O(n(n^2+2n))$ [108]. Такая оценка получена для ускоренной процедуры метода ветвей и границ. Предложенный алгоритм решения *ЛЗН* явно учитывает возможность частичного наследования результатов решения порождающей задачи, «потенциальный эффект реализации такого приема объясняется тем, что вычислительные сложности решения независимых *ЛЗН* — $O(n^3)$, а сложность пересчета после изменения строки матрицы *ЛЗН* — $O(n^2)$. В случае разреженных матриц вычислительная сложность решения отдельных *ЛЗН* снижается пропорционально доле значимых элементов матрицы» [108, с. 26]. Здесь оценка вариантов порожденных задач проводится методом коррекции дерева кратчайших путей приращений. Данный подход встраивается, как элемент, в задачу m коммивояжеров с предварительной кластеризацией исходной сети.

В [103] предложен алгоритм решения *TSP*, в котором:

— методом последовательных сепараций ищется решение задачи о назначениях;

— если найденное решение не является решением *TSP*, то склеивается решение *TSP* из нескольких контуров задачи о назначениях.

2.3.2 Метод максимального разреза

Сложным сетям отвечают графы большой размерности и сложной структуры. Это ограничивает использование классических алгоритмов решения экстремальных задач на графах. Оставаясь в рамках детерминированных моделей, будем использовать особенности (специфику) исследуемых графов: разреженность, когда графы имеют мало ребер по сравнению с числом вершин; «тесноту» фрагментов графа, которым присущ высокий уровень достижимости вершин («тесные миры» — small World Graphs), в этом случае выясняются структурные характеристики графа (кластеризация, средняя длина пути, диаметр кластера и др.)

Определение 2.9. *Связный граф* $G = (V,U)$ *называется разреженным, если число ребер* $|U|$ *удовлетворяет условию*

$$|U| \leqslant \alpha n^{\beta},$$

где $\alpha > 0$, $1 \leqslant \beta < 2$ — *положительные вещественные числа*; $n = |V|$.

Чем меньше β, тем разряженней граф G. В работе [10] используется для характеристики разреженности графа понятие древовидной ширины графа $tw(G)$. Если $tw(G) \leqslant k$, где k — некоторое заданное положительное число, $k < n = |V|$. Чем меньше значение k, тем более разреженным является граф G. Для разреженных графов предлагается [10] декомпозиционный подход, состоящий в сведении решаемой задачи для разреженного графа большой размерности к конечному множеству задач для графов меньшей размерности. При

этом граф разбивается на конечное множество атомов $\Omega(G)$ — атомарное представление графа. После решения задачи на каждом атоме осуществляется построение решения задачи для исходного графа путем корректного попарного соединения решений, полученных для атомов. Указывается, что декомпозиционный подход приводит к новым алгоритмам, позволяющим обрабатывать разреженные графы большой размерности за реальное время.

Атомарное представление применено к задаче вычисления кратчайших путей для всех вершин взвешенного графа (All-Pairs Shortest-Path) совместно с использованием алгоритма Флойда—Уоршелла [87], что позволило найти точное решение за время $O(nk^3)$. Совместное применение атомарной декомпозиции и алгоритма Уилфа [87] к каждому атому и связывание решений, полученных для каждого атома входного графа, приводит к точному решению задачи нахождения наибольшей клики (Maximum-Clique-Problem) за время $O(n \cdot 1{,}39^k)$.

В случае $mTSP$ аналогом данного подхода является разбиение сложной сети на кластеры, на каждом из которых решается своя задача коммивояжера. Адаптивную кластеризацию можно осуществлять различными способами.

Для задач $mTSP$ эффективной технологией является применение алгоритмов решения задачи построения *MAX-CUT* в условиях априорной информации, знаний о моделях. Применение *MAX-CUT* позволяет также формировать базы прецедентов, моделей, знаний (продукционного типа, четких и нечетких), начиная от простых случаев, малой размерности. *MAX-CUT* также используется в задаче кластеризации.

Отметим возникающую при таком подходе кластерную задачу коммивояжера, в которой рассматривается сеть из нескольких кластеров (см. раздел 1.1). Данная задача возникает также при решении многокритериальных задач для многих коммивояжеров. Каждый кластер представляет собой локальную сеть; т. е. граф $H_i = (I_i, U_i)$, $i = 1,2,\ldots,m$, элементам которого приписаны некоторые величины (интенсивности, расстояния, стоимости, затраты, время и т. д.). Путь между вершинами k и l кластера H_j может удовлетворять различным требованиям: быть гамильтоновым, выполнять срединные условия [79], удовлетворять нескольким критериям и т. п. Коммивояжеры могут конкурировать между собой (игровые ситуации) или кооперироваться, сотрудничать и преследовать одну общую цель. Гамильтонов контур может проходиться в виде эстафеты несколькими коммивояжерами. Отметим, что определение целесообразного количества коммивояжеров зависит от множества критериев и дополнительной информации, а также от предпочтений *ЛПР* в условиях реализации метода существующих запросов и экспертных оценок. Для решения таких задач применяются композиции точных, приближенных и эвристических алгоритмов.

Приведем постановку задачи *MAX-CUT* [75]. Пусть задан неориентированный граф $H = (I,U)$ с множествами вершин I и ребер U. Каждому ребру $(i,j) \in U$ графа соответствует вес $w_{ij} > 0$. Разрезом графа H называется разбиение (I_1, I_2) множества его вершин I на два непересекающихся подмножества I_1 и I_2, такие, что $i \in I_1$, $j \in I_2$. Любое такое разбиение порождает разрез графа.

Задача о максимальном взвешенном разрезе неориентированного графа ($MAX\text{-}CUT$) H состоит в нахождении разреза максимального суммарного веса

$$w(I_1, I_2) = \sum_{i \in I_1, j \in I_2, (i,j) \in U} w_{ij}.$$

Задача является NP-трудной, поэтому для больших размерностей эффективны только приближенные методы.

$MAX\text{-}CUT$ используется в решении $mTSP$. Предусматривается наполнение базы моделей, алгоритмов, прецедентов, начиная с малых размерностей m и n. Исходный граф $H = (I,U)$ разбивается на два графа $H_i = (I_i, U_i)$, $i = 1,2$ и, если $m = 2$, то выбор алгоритма решения зависит от размерности $n_i = |I_i|$, $i = 1,2$ каждого подграфа, а также от множества критериев $J = J_0 \cup \ldots \cup J_m$, где J_0 — критерии для всей сети и всех коммивояжеров, J_i — индивидуальные критерии j-го коммивояжера, $j = 1,2,\ldots,m$. Выделяются методы решения задачи для $m = 1$, $J = J_0 \cup J_1$. Дальнейшее решение задачи определяется согласованной или конкурентной работой коммивояжеров (захват сети). Целесообразное количество коммивояжеров m определяется в зависимости от множества критериев J и дополнительной информации, а также от предпочтений $ЛПР$ в условиях реализации метода существующих запросов и экспертных оценок. В случае $m > 2$ разбиение подграфов проводится в соответствии с методом ветвей и границ.

В задаче кластеризации выделение H_k, $k = 1,2,\ldots,K$ зависит от множества критериев J.

Выводы

1. Выделено новое направление — построение многоагентных задач многих коммивояжеров на базе моделей псевдобулевой оптимизации с ограничениями в виде дизъюнктивных нормальных форм. Теоретически обосновано представление $mTSP$ в виде модели псевдобулевой оптимизации с $ДНФ$ ограничениями.

2. Построены алгоритмы решения $mTSP$ в одно- и многокритериальных постановках.

3. Показано, что процедура поиска и построения решения на базе логического выбора является полиномиальной.

4. Предложена общая схема решения задачи $mTSP$ с выделением кластера для каждого коммивояжера и с последующим решением задачи псевдобулевой оптимизации с $ДНФ$ ограничением на каждом кластере с помощью полиномиального алгоритма.

5. Обосновано, что построение рациональных решений $mTSP$ на сетях большой размерности реализуется по схеме алгоритма $AmTSP$ для m коммивояжеров.

6. Показана важность использования постоптимального анализа (реоптимизация) в задачах $mTSP$.

7. Для разреженных сетей или сетей с дугами существенно разного веса рационально применение алгоритмов максимального разреза для схемы кластеризации сети и последующего поиска с оптимальных маршрутов $mTSP$.

Глава 3. Прикладные алгоритмы маршрутизации

3.1 Алгоритмы решения задач маршрутизации с учетом имеющейся информации

3.1.1 Выбор алгоритмов приближенного решения задачи коммивояжера

Задачи маршрутизации. В прикладных задачах представляют интерес процессы, происходящие на графах: перемещение объектов, потоки ресурсов [57], транспортные потоки, информационные потоки, логистика потоков; экономические, социальные, политические, сетевые задачи управления и др. Прикладная теория графов (или алгоритмическая теория графов) связана с построением различного рода путей, маршрутов, распределением потоков, разбиением, кластеризацией и декомпозицией графов. Моделирование в таких задачах тесно связано с учетом свойств структуры сети, ее сложности, наличия ограничений, предписаний, условий достижимости (см., например, работы А. Н. Сесекина и соавторов [117; 118], Е. Е. Иванко [63], Я. М. Ерусалимского [41]). Учет таких знаний, с одной стороны, позволяет получать более адекватную практическую модель задачи, с другой стороны, требует адаптации или разработки аналогов классических алгоритмов [78] решения задач дискретной оптимизации. Задачи маршрутизации являются важной составляющей прикладной теории графов и дискретной оптимизации *ДО*.

Фундаментальные результаты по вопросам теории графов, связанных с построением приближенных алгоритмов для задачи $mTSP$ и других близких задач маршрутизации, получены в работах различных авторов. Полиномиальные приближенные алгоритмы с гарантированными оценками точности для различных вариантов задач $mTSP$ анонсированы А. А. Агеевым, А. Е. Бабуриным, Э. Х. Гимади, А. Н. Глебовым, А. В. Гордеевой, Д. Ж. Зембалаевой, Н. М. Коркишко, А. В. Пяткиным, О. Ю. Цидулко и др. [93]. Для разработки таких алгоритмов применяется подход построения гамильтоновых циклов из наборов вершинно непересекающихся цепей, покрывающих все вершины графа. Наиболее часто вспомогательными структурами являются паросочетания, цикловые покрытия, регулярные подграфы и деревья, связанные k-факторы. Для задачи о связном 2-факторе метрической задачи о связном k-факторе на минимум известен полиномиальный p-приближенный алгоритм, где p — константа [150].

В этой главе рассматриваются алгоритмы задач прикладной маршрутизации. Сравниваются, в частности, эвристические алгоритмы решения задачи поиска кратчайшего пути и задачи типа m коммивояжеров в случае наличия дополнительной информации [19]. Такая информация меняет математическую постановку задачи и алгоритмы ее решения.

В построении приближенных алгоритмов *ДО* в основном используются эвристики, в которых учитываются априорные, прецедентные знания о решении, структуре задачи. Важной

является задача определения насколько приближенное решение отличается от оптимального. Для такой оценки применяют ряд методов:

1. Эмпирическое сравнение. На репрезентативной выборке (например, *TSPLIB* [247]) тестовых задач сравнивается работа различных алгоритмов. В качестве критериев сравнения используются качество полученного решения, время вычисления, трудоемкость. Эмпирические сравнения позволяют выбрать алгоритмы, которые являются эффективными в прикладных задачах.
2. Анализ наихудшего случая. Эвристики могут хорошо работать на большинстве прикладных задач и очень плохо в частных случаях. Алгоритм с лучшей оценкой для наихудшего случая будет эффективным в прикладных задачах (алгоритмы гарантированного функционирования).
3. Анализ среднего случая. Данный подход применяют для большой размерности сложных сетей с известными вероятностными характеристиками. Оценивается эффективность эвристического алгоритма (относительная погрешность решения) при некоторых предположениях о распределении входных данных. Для задач $mTSP$ на сложных сетях представляет интерес информация о распределении длин дуг (в виде гистограмм).

Сложность сетей и задач на графах требует сочетания локальных, эвристических, метаэвристических алгоритмов, построения максимальных разрезов; кластеризации, учитывающей структуру графа.

Как правило, в реальных задачах применяется сочетание методов.

Для построения начальных допустимых решений удобно применять жадные алгоритмы (хотя они могут приводить к заведомо плохим решениям). Например, в задаче построения минимального остовного дерева жадные алгоритмы гарантируют получение оптимального решения (для неориентированного связного графа строится связный ациклический граф, содержащий все вершины графа минимального веса). Решение может быть найдено с помощью алгоритма Прима ([78]).

Для *TSP* в жадном алгоритме осуществляется переход в ближайшую вершину на каждом шаге. Лучший вариант состоит в многократном применении жадного алгоритма для различных начальных вершин. Решение *TSP*, построенное с помощью жадных алгоритмов, может как угодно сильно отличаться от оптимального. Для практического использования этих алгоритмов важна идентификация (распознавание) таких случаев плохих задач.

Для эвристического алгоритма гарантированного функционирования оценка максимального отклонения от оптимального решения определяется следующим образом: пусть $f^*(I)$ и $f(I)$ соответственно оптимального и решения, полученного с помощью эвристики на входной информации I, тогда

$$R = inf\left\{r \geqslant 1, \frac{f(I)}{f^*(I)} \leqslant r,\ \forall I\right\}$$

абсолютная оценка эвристики. Оценка в худшем случае не более чем в R раз превосходит оптимальную. Асимптотическая оценка для большой размерности определяется следующим

образом:

$$R_\infty = inf\left\{r \geqslant 1 : \exists n, \frac{f(I)}{f^*(I)} \leqslant r,\ \forall I \text{ и } f^*(I) \geqslant n\right\},$$

где $R_\infty \leqslant R$. Относительная погрешность алгоритма гарантированного функционирования: пусть $\varepsilon = R - 1 > 0$,

$$\frac{f(I) - f^*(I)}{f^*(I)} \leqslant \varepsilon.$$

Соответствующая эвристика называется ε-оптимальной.

Принятый в работе подход алгоритмической теории построения маршрутов базируется как на псевдобулевой условной оптимизации (глава 2), так и на комбинаторной оптимизации, геометрическом представлении, методах распознавания образов, методах машинного обучения и алгоритмах интеллектуализации обработки данных. При этом, как правило, считается, что задача существования, единственности и устойчивости решения в какой-то мере уже решена. Приведем еще одну характерную постановку, предполагающую существование решения. Вопросы существования и единственности остаются открытыми.

Задача 2 *TSP*. Известно, что для сети $\langle G(V,U), C\rangle$ существует только два гамильтоновых контура, покрывающих вершины графа $G(V,U)$ без пересечений. Необходимо построить классифицирующую функцию, относящую вершины из G к одному из контуров (или позволяющую построить разрез).

Для алгоритма достаточно указать две вершины, относящиеся к двум контурам и процедуру построения остальных. При этом в решении NP-полной задачи $mTSP$ участвуют полиномиальные алгоритмы или сама задача является полиномиальной.

Алгоритмы прикладных задач маршрутизации. Для задач *ДО* на графах существуют хорошие алгоритмы в случае планарных, метрических графов с весовыми коэффициентами, удовлетворяющими, например, неравенству треугольника $l_{ij} \leqslant l_{ik} + l_{kj}$, $l_{ij} \geqslant 0$.

В реальных задачах между вершинами заданы расстояния $c_{ij} \geqslant 0$, где $(i,j) \in U$ — множество дуг, $i,j \in V$ — множество вершин. Зная координаты вершин $i \in V$, $|V| = n$, можно поставить расстояниям c_{ij}, не удовлетворяющим неравенству треугольника, расстояния $l_{ij} = \rho(i,j)$ равные расстоянию по прямой между вершинами i и j. Такого типа задачи возникают для задач облета вершин (объектов) на некоторой высоте над поверхностью. Понятно, что в общем случае $c_{ij} \geqslant l_{ij}$, а некоторым маршрутам нельзя поставить в соответствие маршруты на поверхности (более короткий путь может лежать через море, горы и другие препятствия). Система запретов и предписаний позволяет выбирать серию модельных графов (даже меньшей размерности), допускающих построение приближенных решений за приемлемое время. Для абстрактных сложных сетей построить отображение (обратимое), сводящее исходную задачу к более простой, не всегда возможно. Это видно на примере программ визуализации сложных сетей. С другой стороны, примеры визуализации указывают на возможность адаптивного, практического, корректного упрощения исходной графовой структуры и решения сложных задач (кластеризации, а тем самым и многоагентной маршрутизации). Такой подход применен для выбора туристических маршрутов. Предлагается следующий обобщенный алгоритм.

Алгоритм 3.1. *Алгоритм прикладной маршрутизации на приведенных сетях*

Вход: *исходный граф* $G(V,U)$ *и весовая матрица* C.

Выход: *приближенное решение задачи ДО на графе* $G(V,U)$.

1: *Задать граф* $G(V,U)$ *и весовую матрицу* C.

2: *Найти преобразование* $\rho : C \to L$, *т. е. по матрице построить матрицу* L *с элементами* $l_{ij} \geqslant 0$, $(i,j) \in U$, *удовлетворяющими неравенству треугольника.*

3: *Учесть априорную информацию, запреты и предписания*; *преобразовать матрицу* L *в матрицу* $\tilde{L}$, *учитывающую данную информацию.*

4: *Провести анализ и упрощение матрицы* $\tilde{L}$ (*метрические характеристики*; *структурные составляющие*: *мосты, сочленения, висячие вершины*; *необходимость кластеризации*), *сформировать упрощенную матрицу* $\tilde{\tilde{L}}$.

5: *Для упрощенной матрицы* $\tilde{\tilde{L}}$ *решить задачу ДО.*

6: *Построить обратное соответствие* $\tilde{\tilde{L}} \to \tilde{L} \to C$ *и получить вариант решения, проверить на соответствие.*

7: *Предъявить приближенное решение исходной задачи ДО.*

Выделение при некоторых упрощениях из класса NP-полных задач маршрутизации экземпляров полиномиальных задач (см. раздел 1.1.1) позволяет строить алгоритмы приближенного решения. Можно поставить задачу: по эвристическому алгоритму решения задачи маршрутизации восстановить постановку исходной задачи, для которой данный алгоритм является точным. Такой систематический подход позволит более точно классифицировать математические модели и алгоритмы маршрутизации (предложение автора).

Для выбора алгоритма нахождения приближенного решения задачи $mTSP$ базовой является классическая TSP: во взвешенном полном неориентированном графе требуется найти гамильтонов цикл минимального веса. Для ориентированного графа, в котором веса прямой и обратной дуг могут не совпадать, требуется найти гамильтонов контур минимального веса.

С этой задачи началось развитие направления комбинаторной оптимизации. Многие методы тестируются на этой задаче. TSP является важной составляющей многоагентных моделей дискретной оптимизации (псевдобулевой однокритериальной и многокритериальной оптимизации, см. главу 2).

Приведем некоторые известные, но необходимые факты, теоремы 3.1–3.5, алгоритмы 3.2–3.5 и сделаем из них выводы, полезные для разработки алгоритмов решения $mTSP$ [56].

Прежде чем искать гамильтоновый цикл в графе, нужно выяснить, является ли данный граф гамильтоновым. Эта задача является NP-полной, а также задачей распознавания в отличие от TSP. Задачи, находящиеся вне класса NP-полных, но не проще этих задач называются NP-трудными.

Теорема 3.1. *TSP является* NP*-трудной, даже в случае симметричности матрицы весов* (c_{ij}=c_{ji}) *и* $c_{ij} \in \{1,2\}$.

Задача является NP-трудной в сильном смысле, если она остается NP-трудной даже при унарной кодировке исходных данных, т. е. когда все числа в исходных данных ограничены некоторой константой.

TSP является NP-трудной в сильном смысле. Из этого следует, что *построить точный полиномиальный алгоритм невозможно*. Это утверждение справедливо и для приближенных алгоритмов.

Пусть для экземпляра I задачи коммивояжера $Opt(I)$ задает вес кратчайшего гамильтонового цикла. Пусть алгоритм A на этом экземпляре дает ответ $A(I)$.

Теорема 3.2. *([56]) Если существует полиномиальный алгоритм A и константа $r > 0$ такая, что для любого экземпляра TSP I справедливо неравенство $A(I) \leqslant r \cdot Opt(I)$, то классы P и NP совпадают.*

Доказательство. Рассмотрим матрицу весов графа $G(V,E)$, построенную следующим способом:

$$c_{ij} = \begin{cases} 1, \ (i,j) \in E; \\ n \cdot r, \ (i,j) \notin E. \end{cases}$$

Если $A(I) = n$, то граф является гамильтоновым (содержит гамильтонов цикл).

Предполагая $A(I) > n$, получаем, что $A(I) \geqslant n \cdot r + (n-1)$, т. к. цикл проходит хотя бы через одно ребро $(i,j) \notin E$, вес которого равен $n \cdot r$. В этом случае граф не может иметь гамильтоновый цикл, т. к. алгоритм ошибается в r раз. То есть полиномиальный алгоритм A дает правильный ответ для NP-полной задачи ($P = NP$), что невозможно. □

Из этого доказательства следует, что полиномиальный алгоритм в худшем случае (если $P \neq NP$) не может быть ограничен даже экспоненциальной величиной $2^{p(n)}$, где $p(n)$ — произвольный полином, n — размерность задачи. Для этого величину r нужно взять $r(n) = 2^{p(n)}$ [209].

Исходя из этого, сделаем следующий вывод: необходимо выделять такие классы *TSP*, для которых найдутся приближенные полиномиальные алгоритмы. Для работы представляют интерес несколько классов таких задач:

— класс *TSP* с весами, удовлетворяющими неравенству треугольника $c_{ij} \leqslant c_{ik} + c_{kj}$, $1 \leqslant i, j, k \leqslant n$ и перечисленные в разделе 1.1.1;

— класс *TSP*, для которых применимы итерационные алгоритмы локального улучшения;

— класс *TSP*, для которых применимы алгоритмы реоптимизации;

— класс *TSP*, для которых применимы конструктивные алгоритмы, метаэвристики и др.

На каждом из направлений возникают свои трудности для возможности применения и эффективной реализации. При этом очевидна трудность построения критериев принадлежности *TSP* к тому или иному классу.

Рассмотрим итерационные алгоритмы локального улучшения.

На каждой итерации используется полиномиальный алгоритм локального улучшения, при этом число итераций может быть экспоненциальным.

Рассмотрим множество циклов графа G. Выберем гамильтоновый цикл Γ и его окрестность $N(\Gamma)$, т. е. все циклы, которые можно получить из цикла Γ локальными преобразованиями. Например, окрестности $N(\Gamma)$ определяется 2-заменой (2-Opt), в которой два несмежных ребра заменяются на два других, приводящих к гамильтоновому циклу. Мощность таких преобразований $n(n-3)/2$. Схема такого алгоритма имеет вид

Алгоритм 3.2. *Итерационный алгоритм локального улучшения A*

Вход: *множество гамильтоновых циклов* Γ *графа* G.

Выход: *цикл минимального веса.*

1: *Выбрать начальный цикл* Γ_0, *найти его вес* $\rho(\Gamma_0)$.

2: *Найти ближайшего соседа* Γ_1 *для цикла* $\Gamma_0 : \Gamma_1 = argmin\{\rho(\Gamma_0), \Gamma_0 \in N(\Gamma)\}$.

3: *Если* $\rho(\Gamma_1) < \rho(\Gamma_0)$, *то* $\Gamma := \Gamma_1$ *и перейти на шаг* 2, *иначе* — $STOP$, Γ — *локальный минимум.*

Теорема 3.3. *Пусть алгоритм локального улучшения A и окрестность* $N(\Gamma)$ *имеет полиномиальную мощность. Если* $\exists r > 0$ *такая, что для любого экземпляра I TSP справедливо неравенство* $A(I) \leqslant r \cdot Opt(I)$, *то* $P = NP$.

Доказательство. Аналогично теореме 3.2. □

Вывод: таким образом, нужно уметь строить допустимые гамильтоновые циклы (разумные с практической точки зрения, компромиссные). С алгоритмической точки зрения для успешной программной реализации они должны быть конструктивными. Такие алгоритмы обладают как достоинствами, так и недостатками. Они не обеспечивают полиномиальную разрешимость *TSP*.

Под *конструктивными алгоритмами* понимают алгоритмы, в которых шаг за шагом строят допустимое решение задачи, например, добавляя одно ребро за другим. Таким алгоритмом является жадный алгоритм, в котором на каждом шаге осуществляется переход в ближайшую из непройденных вершин.

Алгоритм 3.3. *Жадный алгоритм перехода в ближайшую вершину A*

Вход: *сеть* $S = (G,C)$, $G = (V,E)$, $n = |V|$, C — *матрица весов.*

Выход: *допустимый гамильтонов цикл.*

1: *Выбрать произвольную вершину* i_1 *и пометить ее.*

2: *Цикл по* $k = 1, \ldots, n-1$ *найти* i_{k+1} — *ближайшую из непомеченных вершин к вершине* i_k: $c_{i_k i_{k+1}} = \min\limits_{j \neq i_1, \ldots, i_k} c_{i_k j}$ *и пометить вершину* i_k.

Теорема 3.4. *Для* $\forall r > 1$ *найдется экземпляр TSP с информацией I такой, что при выполнении неравенства треугольника справедливо неравенство* $A(I) > r \cdot Opt(I)$.

Теорема 3.5. *Для любого экземпляра TSP I с симметричной матрицей C, удовлетворяющей неравенству треугольника, алгоритм* $A_{E\Gamma}$ *получает гамильтонов цикл* $A_{E\Gamma}(I) \leqslant 2 \cdot Opt(I)$. *Оценка точности* $r = 2$ *является неулучшаемой для этого алгоритма.*

Алгоритм 3.4. *Алгоритм гарантированной точности $r = 2$ [3]*

Вход: *сеть $S = (G,C)$, G — полный взвешенный граф, C — матрица весов.*

Выход: *гамильтонов цикл.*

1: *Построить с помощью алгоритма Краскала A_k остовное дерево минимального веса. $A_k(I) \leqslant Opt(I)$ для любого экземпляра TSP I, т. к. удаление ребра оптимального решения TSP дает остовное дерево.*

2: *Построить эйлеров цикл, предварительно заменив каждое ребро остовного дерева на два ребра.*

3: *Перестроить эйлеров цикл в гамильтонов с помощью алгоритма $A_{EГ}$.*

Утверждение теоремы 3.5 справедливо для перестройки любого эйлерового цикла в гамильтоновый. Если из множества эйлеровых циклов (их может быть экспоненциально много) выбрать наилучший экземпляр, то тогда можно уменьшить ошибку. Такая оценка получена в работе [171]. Поведение алгоритма в среднем дает отклонение нижней оценки не более 10%.

Алгоритм 3.5. *Алгоритм A_{KC} Н. Кристофидеса и А. Сердюкова*

Вход: *G — взвешенный граф, C — симметричная матрица весов, удовлетворяющая неравенству треугольника.*

Выход: *гамильтонов цикл.*

1: *Построить остовное дерево минимального веса.*

2: *Добавить ребра и получить эйлеров граф. Вместо дублирования ребер в остовном дереве выделяется множество вершин V' нечетной степени, $|V'| = 2K$ (т. к. сумма всех степеней вершин графа должна быть четной).*

3: *На множестве V' найти совершенное паросочетание минимального веса: k ребер, имеющих минимальный суммарный вес и покрывающие все вершины (эта задача полиномиально разрешима [102], вес такого паросочетания не превосходит половины длины любого гамильтонового цикла).*

4: *Добавить полученное паросочетание к остовному дереву. Получим эйлеров граф (его вес не превосходит 3/2 длины минимального цикла).*

5: *Построить эйлеров цикл.*

6: *Перестроить эйлеров цикл в гамильтонов.*

Оценку в 3/2 нельзя улучшить. Выбор наилучшего эйлерового цикла является NP-трудной задачей. На основании приведенных известных результатов и выводов по алгоритмам сформулируем следующее следствие.

Следствие 3.1. *С точки зрения учета имеющейся информации справедливы утверждения: для построения приближенного решения TSP используются различные алгоритмы и структуры: эйлеровы циклы, остовные деревья минимального веса, совершенные паросочетания и др. Выбор алгоритмов зависит от исходных знаний о задаче, а также идентификации хороших и плохих экземпляров задач.*

Например, когда вершины графа G расположены на окружности через равное расстояние (веса равны) жадный алгоритм дает не лучший результат. Но обнаружение в графе

структуры моста приводит к оптимальному решению. Таким образом, необходимо уметь находить мосты, точки сочленения, висячие вершины и другие характерные структуры в сети (такие алгоритмы являются полиномиальными или даже линейными).

Покажем, что поиск глобального решения задачи $mTSP$ в рамках *ДО* может осуществляется в рамках итерационного процесса с помощью алгоритмов локального улучшения. Используется понятие окрестности соседства близости решений. В задаче *ДО* $z(f,\Omega)$ задается входная информация (исходные данные) $I(f,\Omega)$, определяется конечное множество решений $S = S(I)$, $I \in I(f,\Omega)$, целевая функция $f : S \to \mathbb{R}$. Здесь Ω — множество, на котором ищется решение задачи $\min\limits_{x\in\Omega} f(x)$. Требуется найти как минимум функции f на множестве допустимых решений $S(I)$, так и само решение $s^* \in S^* \subset S(I)$. Для $\forall s \in S(I)$ определяется функция окрестности $N : S(I) \to 2^{S(I)}$, которая для каждого допустимого решения задает множество соседних допустимых решений. Близость между решениями определяется по-разному и зависит от конкретных условий задачи. $N(s) \subset S(I)$ — окрестности решения в множестве $S(I)$.

Определение 3.1. *([56]) Решение $\hat{s} \in S(I)$ называется локальным минимумом по отношению к множеству $N(s)$, если $\forall s \in N(s)$ выполняется $f(\hat{s}) \leqslant f(s)$. Множество локальных минимумов обозначим $\hat{S}$. Если $\hat{S} \subset S^*$, то функция окрестности N называется точной.*

С помощью алгоритмов локального улучшения на основе имеющейся информации осуществляется поиск глобального минимума, который не улучшаем с помощью локальных процедур.

Переход от одного допустимого решения к другому с помощью локальных улучшений задает некоторый итерационный процесс. Для перехода необходим выбор направления и шага. Понятия направления и шага в каждом конкретном случае требуют уточнения в зависимости от структуры $N(s)$ и близости допустимых решений. Близость или соседство задается.

Определение 3.2. *([56]) Графом соседства $G_N = (S(I), E)$ называется взвешенный граф, вершинами которого являются допустимые решения задачи, а дугами — упорядоченные пары (s,t), если $t \in N(s)$. Веса приписываются вершинам и равны значениям целевых функций.*

Граф соседства G_N (neighborhood graph) задает ландшафт целевой функции f (landscape, fitnesslandscape).

Для локального улучшения (переход от одного допустимого решения к другому — лучшему) необходима связность графа G_N, т. е. для каждой пары (s,t) должен существовать путь из s в t. Рассматривают свойство вполне связности, для которого существует путь из любой вершины в вершину $s^* \in S^*$. Отсутствие таких свойств приводит к необходимости переопределения $N(s)$.

Для *TSP* алгоритмы локального поиска на практике оказываются лучшими. Есть пример окрестностей алгоритма Лина—Кернигана [218], который обеспечивает среднюю погрешность в 2% на 1000000 вершинах [209]. Для случайных графов такой размерности итерационная процедура Джонсона находит решение с 0,5% отклонением за несколько минут [209].

Решение задачи $mTSP$ может быть сведено к решению задачи одного коммивояжера на графе большей размерности или к m задачам TSP на графе меньшей размерности, чем исходный. При этом точные алгоритмы применимы для графов небольших размеров. Эвристические алгоритмы нацелены на приближенное решение, которое может сильно отличатся от точного. Для многих практических задач важно получить точное или максимально близкое к точному. Компромисс достигается на применении композиции различных алгоритмов, сочетании эвристик и точных алгоритмов. Естественным для $mTSP$, TSP является точный алгоритм ветвей и границ, впервые предложенный [147] на основе работы [213]. Одним из подходов по разработке модифицированных точных алгоритмов с хорошей временной эффективностью является комбинация эвристического алгоритма в качестве начального приближения и точного алгоритма ветвей и границ с сокращенным поисковым деревом решений [188; 197; 225].

Наилучшими кандидатами эвристических алгоритмов выбора предварительных решений являются алгоритмы, улучшающие решения типа Лина—Карнегана (LK) [218], основанные на итерационном улучшении решения. Сложность алгоритма порядка $O\left(n^{2,2}\right)$ и он часто позволяет находить глобальное оптимальное решение. Модифицированные версии алгоритма Лина—Карнегана в 2000 году предложены К. Хелсгауном в работе [194], где описана стратегия поиска. В алгоритме Лина—Карнегана—Халсгауна (LKH) находится допустимое решение и выделяется два множества дуг. Дуги первого множества удаляются из допустимого решения и заменяются дугами второго множества. В результате чего решение улучшается. Алгоритм поиска на основе ограничений 5-Opt перемещений и все ограничения на дуги формируемых множеств приводится в [194; 195]. Главная задача комбинации эвристических и точных алгоритмов состоит в ускорении работы комбинированного алгоритма по сравнению с классическим методом ветвей и границ.

Оценка размерности графа для комбинации алгоритмов. Допустимую размерность сети (количество вершин n для полного симметричного графа) для такой комбинации алгоритмов можно оценить на основе экспериментов, в результате которых строится приближенная формула для оценки работы алгоритма. Например, среднее время $\bar{t}_{\text{К}} = \alpha e^{\beta n}$ для комбинированного алгоритма и $\bar{t}_{\text{МВГ}} = \gamma e^{\delta n}$ для метода ветвей и границ. Зная параметры α, β, γ, δ, находим приемлемое максимальное время $\bar{t}_{\text{МВГ}} = \bar{t}_{max}$ и вычисляем n для комбинированного алгоритма. Из уравнения $\alpha e^{\beta n} = \bar{t}_{max}$ получаем $n_{max} = \dfrac{1}{\beta} ln \dfrac{\bar{t}_{max}}{\alpha}$. Конечно, такая оценка является грубой. Для ассиметричных разреженных графов необходима уточненная оценка, которую можно получить в результате многократного тестирования разрабатываемого алгоритма. В работе [58] в результате экспериментальных исследований несимметричных TSP с помощью эвристических алгоритмов и метода ветвей и границ получено соотношение $27{,}5exp(0{,}2293n) = \bar{t}_{max}$ $\alpha = 27{,}5$, $\beta = 0{,}2293$, для $\bar{t}_{max} = 5$мин получено $n_{max} = 73$. Ситуация усложняется, если необходимо решать задачи TSP, $mTSP$ с $n \gg 100$.

Метаэвристики. Рассмотрим важный для $mTSP$ алгоритм табу-поиска (поисковый алгоритм с запретами, tabu search algorithm — TSA), предложенный Фредом В. Гловером в 1986 г., который является метаэвристическим алгоритмом локального поиска. Запреты (tabu)

вводятся, чтобы препятствовать TSA возвращаться к ранее посещаемым решениям. Метаэвристики — это алгоритмы улучшения. Они начинают с одного или нескольких допустимых решений, и далее эти решения улучшаются. К типичным метаэвристикам относятся алгоритмы локального поиска, *TSA*, имитации отжига и *GA*. TSA наиболее широко используемые метаэвристики для дискретной (комбинаторной) оптимизации. В TSP поиск начинается с допустимого решения (текущего) далее ищется лучшее решение в подходящей окрестности (совокупность маршрутов). Затем определяется текущее решение как лучшее в окрестности, и снова начинается поиск. Для того, чтобы *TSA* не учитывал решения, которые посещались на последних итерациях, создается табу-список. Как только вариант решения входит в список табу, такие решения считаются запрещенными при поиске соседних решений. Список табу непрерывно меняется во время выполнения поиска, т. е. алгоритм *TSA* является адаптивным поиском в памяти. Обобщение алгоритма осуществлялось по пути организации среднесрочной и долгосрочной структур памяти и др.

Важным этапом является выбор первоначального маршрута TSP. Рассмотрим некоторые *алгоритмы генерирования начальноо решения*.

Модификацией жадного алгоритма является алгоритм случайного адаптивного поиска. Обход строится путем увеличения маршрута в графе и соединения конечных точек, как только все узлы включены в маршрут. Первая вершина выбирается случайным образом. Точка, которая добавляется к растущему пути, не обязательно является ближайшей к конечным точкам, а случайной, выбранной из набора точек, которые достаточно близки к конечным точкам существующего пути.

Случайная вставка (Randomized Insertion — *RandIns*) начинается с частичного маршрута. Случайным образом вставляются узлы в маршрут без промежуточных маршрутов. Алгоритм останавливается, когда все узлы в графе включены в маршрут.

Алгоритм ближайшего соседа начинается с частичного маршрута. Затем ищется узел, не включенный в частичный маршрут и имеющий минимальное расстояние от последнего добавленного узла в частичный маршрут.

Если граф, описывающий *TSP*, не является полным, то существует вероятность, что некоторые узлы не будут включены в итоговый маршрут. Эти узлы добавляются, используя процедуру случайной вставки *RandIns*.

Алгоритм развертки (Sweep heuristic — *Sweep*) применяют для *TSP* на плоскости. Выбирается опорная точка (например, центр масс). Затем находятся вершины в порядке возрастания угла между фиксированной осью, проходящей через опорную точку и определяемой вершиной. Для многих коммивояжеров с несколькими складами склады используются в качестве опорных точек. Данный алгоритм используется в разделе 3.3. Для задач маршрутизации транспортных средств также используются модификации указанных алгоритмов.

Алгоритмы *TSA* отличаются между собой методами перехода от одного решения к другому на базе правил и называются ходами. Достаточно подробный обзор к 2012 г. по *TSA* представлен в работе [154]. Вариант алгоритма для больших графов представлен в [179]. Тестовые задачи для TSP можно найти в [229].

В работах А. Б. Муравника и соавтора [43; 100] TSP ставится в соответствие функционал Ляпунова, по которому формируется искусственная нейронная сеть (*ИНС*). Схема применима для решения $mTSP$. Сложность замечается в подборе параметров нейронной сети, который связан с процессом обучения *ИНС*.

3.1.2 Маршрутизация с ограничениями

Задача построения кратчайшего маршрута в взвешенном графе, как и TSP (глава 2), сводится к задаче псевдобулевой оптимизации с *ДНФ* ограничениями. Если заданы начальная, конечная вершины и необходимо посетить все промежуточные вершины, то приходим к TSP.

В отличие от классических задач нахождения кратчайшего пути, а также маршрута коммивояжера и т. п. содержат ограничения по прохождению вершин, дуг подмножеств графа сети, различных дополнительных условий. Такими алгоритмами могут оснащаться интеллектуальные агенты (*ИА*), которые совместно решают задачи маршрутизации в сложных сетях.

Для решения подобных задач традиционно рассматриваются детерминированные методы, как, например, метод касательных или метод Дейкстры, гарантирующие нахождение глобального оптимума, а также другие подходы к поиску кратчайшего пути [62]. В общем случае использование точных методов неэффективно и неперспективно.

В связи с этим для решения оптимизационных задач практической маршрутизации целесообразнее разрабатывать и использовать эвристические методы. Они позволяют найти субоптимальное, а иногда и оптимальное решение за разумное время.

В работе [64] исследуются подходы к решению задачи нахождения эффективного (кратчайшего) пути в двумерном пространстве с препятствиями. В качестве инструментов решения данной задачи рассмотрены эвристические методы: *муравьиный алгоритм* (*МА*), *метод роящихся частиц* (*МРЧ*) и *эволюционно-генетический алгоритм* (*ЭГА*). Исследованы модификации алгоритмов этих методов, адаптированные для решения поставленной задачи. В случае использования *МРЧ* показано, что ближе всего к рассматриваемой задаче подходит гибрид элементов *поведенческих моделей роя, стаи птиц и ряда других вариантов мультиагентного взаимодействия*. При исследовании *МА* выявлено, что удобно применять классическое разбиение пространства поиска на несопоставимо мелкие по сравнению с препятствиями фрагменты. Как указывалось ранее, агенты-муравьи используют традиционную логику выбора перехода из фрагмента во фрагмент: память о наиболее популярных маршрутах на основе феромона, а также элементы тактики принятия и ситуационно обоснованных, и случайных решений. Установлено, что это дает эффективное улучшение маршрута, а также использование *ЭГА* имеет значительные трудности, связанные с особенностями операторов формировании популяции, кроссинговера и мутации. Доказано, что метод *ЭГА* уступает *МРЧ* и *МА* по эффективности в этой задаче. В отличие от задач обхода препятствий на

плоскости *ИА* агенты-коммивояжеры совершают передвижение по дугам графа. Необходима адаптация алгоритмов решения задач маршрутизации при наличии дополнительных ограничений. При этом нужно:

— отобрать алгоритмы нахождения кратчайшего пути на взвешенном графе (веса — расстояния), пригодные для дальнейшего использования в задачах построения кратчайшего пути с ограничениями;

— сформировать набор задач, в которых учитывается прохождение выбранной дуги (или запрет на прохождение); запреты на посещение вершин; условия посещения группы вершин или дуг (или запретов);

— выяснить применимость алгоритмов (с ограничениями) для решения более сложных задач маршрутизации — типа многих коммивояжеров.

Приведем набор задач, постановка которых определяется наличием тех или иных ограничений, допускающих формализацию ограничений в виде *ДНФ*. Показано, как изменяется классический алгоритм нахождения кратчайшего пути при наличии ограничений на дуги, вершины или их сочетания. Предложенные задачи и алгоритмы частично отражают более общую программу исследований задач маршрутизации типа коммивояжера и описаны в работах автора [18; 20; 23; 29; 74].

1. Нахождение кратчайших путей на графе с дополнительными условиями. Пусть дан ориентированный граф $G(V,U)$, где V — множество вершин, $|V| = n$, U — множество дуг; $i,j \in V$ и матрица расстояний (весов) C с элементами $c_{ij} > 0$, $(i,j) \in U$, если $c_{ij} = \infty$, то дуга (i,j) — отсутствует. Следуя работам автора [20; 23; 26], конкретизируем алгоритмы поиска кратчайших путей в графе с дополнительными ограничениями.

Задача 3.1. Нахождение кратчайших путей из вершины $s \in V$ в вершину $t \in V$ (или между всеми вершинами).

Алгоритмы решения таких задач приводятся во многих источниках (классические, точные, приближенные, эвристические и т. п.) [3; 4; 66; 78; 83; 111].

Задача 3.2. Требуется найти кратчайший путь $p(s,t)$ из вершины $s \in V$ в вершину $t \in V$ $(s \neq t)$ с дополнительными ограничениями: кратчайший маршрут должен проходить через дугу $(k,l) \in U$. Алгоритм задачи 3.2:

1: *Найти кратчайший путь* $p(s,k)$ *из вершины* $s \in V$ *в вершину* $k \in V$.

2: *Найти кратчайший путь* $p(l,t)$ *из вершины* $l \in V$ *в вершину* $t \in V$.

3: *Сформировать искомый кратчайший путь* $p(s,t) = p(s,k) \bigcup p(k,l) \bigcup p(l,t)$.

Доказательство очевидно.

Рисунок 3.1 — Ограничение на дугу k,l

Задача 3.3. Требуется найти кратчайший путь $p(s,t)$ из вершины $s \in V$ в вершину $t \in V$, не проходящий через дугу (k,l). Алгоритм задачи 3.3:

1: *В матрице* C *назначить дуге* (k,l) *вес* $c_{kl} = \infty$ *(или исключить дугу* $(k,l) \in U$*), получить новую матрицу* $\widetilde{C}$.

2: *На новом графе* $G(V,\widetilde{U})$ *с матрицей* $\widetilde{C}$ *найти кратчайший путь.*

Задача 3.4. Требуется найти кратчайший путь $p(s,t)$ из вершины $s \in V$ в вершину $t \in V$, проходящий через вершину $k \in V$. Алгоритм задачи 3.4 сводится к алгоритму задачи 3.2:

1: *Найти* $p(s,k)$.

2: *Найти* $p(k,t)$.

3: *Найти искомый путь* $p(s,t) = p(s,k) \bigcup p(k,t)$.

Задача 3.5. Требуется найти кратчайший путь $p(s,t)$ из вершины $s \in V$ в вершину $t \in V$, не проходящий через вершину $k \in V$. Алгоритм задачи 3.5:

1: *Удалить вершину* k *из множества вершин* V, *т. е. удалить все дуги, входящие и выходящие из вершины* k. *Или в матрице* C *веса дуг* i, k, k, j *для* $i,j \in V$ *положить равными* $\infty : c_{ik} = c_{kj} = \infty$, $i,j \in V$, $i \neq k$, $j \neq k$ *и получить новую матрицу* $\widetilde{C}$.

2: *На новом графе* $G(V,\widetilde{U})$ *с матрицей* $\widetilde{C}$ *найти кратчайший путь* $p(s,t)$.

Если для исходного графа $G(V,U)$ найдены кратчайшие пути между всеми вершинами из V, то алгоритмы задач 3.2–3.4 начинаются с проверки выполнения ограничений для уже ранее найденного кратчайшего пути $p(s,t)$. Дальнейшее обобщение задач 3.2–3.4 состоит в требовании прохождения кратчайшего пути через несколько дуг или вершин (или запрета).

Задача 3.6. Искомый кратчайший путь $p(s,t)$ должен содержать фрагменты $p_k(i_k,j_k)$, $k = 1,2,\ldots,m$.

Рисунок 3.2 — Фрагмент сети с предписаниями

Пусть таких фрагмента два $p_1(i_1,j_1)$ и $p_2(i_2,j_2)$. Возможны следующие случаи: алгоритм решения задачи 3.6 (вариант 1).

Первый случай.

1: *Найти кратчайший путь* $p(j_2,i_2)$.

2: *Найти кратчайшие пути* $p(s,i_2)$ *и* $p(j_2,t)$.

3: *Найти искомый путь* $p(s,t) = p(s,i_1) \bigcup p_1(i_1,j_1) \bigcup p(j_1,i_2) \bigcup p_2(i_2,j_2) \bigcup p(j_2,t)$.

Второй случай.

1: *Найти кратчайший путь* $p(j_2,i_1)$.

2: *Найти кратчайшие пути* $p(s,i_2)$ *и* $p(j_1,t)$.

3: *Найти искомый путь* $\widetilde{p}(s,t) = p(s,i_2) \bigcup p_2(i_2,j_2) \bigcup p(j_2,i_1) \bigcup p(i_1,j_1)p(j_1,t)$.

4: *Сравнить расстояния* $l = |p(s,t)|$ *и* $\widetilde{l} = |\widetilde{p}(s,t)|$, *и выбрать путь, отвечающий* $min(l,\widetilde{l})$.

Алгоритм задачи 3.6 (вариант 2):

1: *Преобразовать граф* $G(V,U)$ *в граф* $\widetilde{G}(\widetilde{V},\widetilde{U})$ *с новой матрицей* $\widetilde{C}$.

(а) *Совмещаем вершины* i_k *с* j_k, $k = \overline{1,m}$. *Для промежуточных вершин пути* $p_k(i_k,j_k)$ (*т. е. кроме вершин* i_k *и* j_k) *удалить все входящие и выходящие дуги.*

(б) *В новой вершине* i_k *оставить дуги, входящие в* i_k *и выходящие из вершины* j_k.

2: *Найти кратчайший путь, проходящий через вершины* j_k, $k = \overline{1,m}$.

Для преобразованного графа $\widetilde{G}(\widetilde{V},\widetilde{U})$ с матрицей $\widetilde{C}$ возможен и другой алгоритм.

Алгоритм задачи 3.6 (вариант 3):

1: *Найти кратчайший путь* $p(s,t)$. *Если фрагменты* $p_k(i_k,j_k)$, $k = \overline{1,m}$ *принадлежат* $p(s,t)$, *то найденный путь является искомым. Если часть фрагментов* $p_k(i_k,j_k)$, $k = \overline{m_1,m}$ *не принадлежат найденному пути* $p(s,t)$, *то включить их в новый кратчайший путь* $\widetilde{p}(s,t)$.

2: *Использовать алгоритм задачи* 3.8.

Следующие задачи также содержат ограничения на фрагменты, принадлежащие к оптимальному маршруту или не принадлежащие.

Задача 3.7. Искомый кратчайший путь $p(s,t)$ не должен содержать фрагменты $p_k(i_k,j_k)$, $k = \overline{1,m}$.

Описание алгоритмов задач 3.7, 3.8, 3.9 приводится аналогично предыдущим.

Задача 3.8. Искомый кратчайший путь $p(s,t)$ должен содержать фрагменты $p_k(i_k,j_k)$, $k = \overline{1,m_1-1}$ и не должен содержать фрагменты $p_k(i_k,j_k)$, $k = \overline{m_1,m}$.

Задача 3.9. Для множества вершин $V_s \subset V$ и $V_t \subset V$ найти кратчайшие пути $p(s,t)$ между вершинами множества $s \in V_s$ и всеми вершинами $t \in V_t$. Найти множества общих вершин и общих дуг для путей $p(s,t)$, $s \in V_s$, $t \in V_t$. Построить распределения для числовых характеристик.

Приведенный список задач и алгоритмов является частью более общей программы. Рассмотрим реализацию алгоритма. Программа допускает изменения для решения других задач.

2. Программная реализация для задачи нахождения всех кратчайших путей между заданной парой вершин с учетом дополнительных ограничений. В качестве ограничений рассматриваются запреты на посещение вершины и на посещение ребра.

Программа представлена в виде двух основных модулей. В первом модуле на вход подается единственный параметр — номер начальной вершины. Стартуя из указанной вершины, алгоритм модуля находит кратчайший путь до каждой из вершин графа. Во втором модуле на вход подаются два параметра — номера начальной и конечной вершин. Стартуя из указанной вершины, алгоритм модуля находит количество кратчайших путей до каждой из вершин графа.

В основном модуле считывается заданный пользователем граф в следующем формате: количество вершин, номер стартовой вершины, номер конечной вершины, особый символ для вывода (‘/’ или ‘%’). Далее считываются количество дуг из i-ой вершины, а далее номер j-ой вершины и вес $w_{ij} = c_{ij}$ дуги.

Рассмотрим модельный граф, состоящий из 12-ти вершин. Пусть необходимо найти кратчайший путь из вершины 0 в вершину 7.

Существует три пути, каждый из которых обязательно пройдет через вершины 4 и 5. Длины этих путей будут равны 14. После введения ограничения на невозможность посещения вершины с номером 5 будет получен единственный путь длиной 16. Теперь рассмотрим ситуацию, когда дан запрет на прохождение по дугам (0, 4) и (3, 6). В этом случае снова получим три пути, но теперь они будут обязательно проходить через вершины 1, 2, 3 и 5.

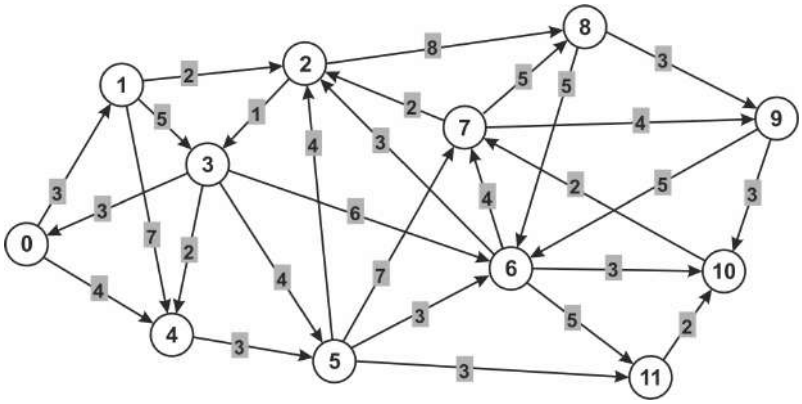


Рисунок 3.3 — Граф задачи, состоящий из 12 вершин

Во втором модуле проводится подсчет количества путей. Принцип его работы в точности совпадает с описанным ранее, но теперь алгоритм работает не со всеми дугами. Маршрут будет проходить только такие дуги, что кратчайший путь от стартовой вершины до заданной проходит через дугу (i, j). Здесь присутствует такое же условие, устраняющее из рассмотрения недоступные дуги. Проверять переход в запрещенную вершину нет необходимости, поскольку она будет помечена первым модулем как недостижимая, а значит, не существует оптимального пути в эту вершину. Проверка дуги обусловлена реализацией программы, поскольку в действительности дуги не устраняются из общего списка, а просто помечаются. Тогда есть вероятность, что дуга (i, j) будет иметь вес, позволяющий попасть в j-ую вершину с минимальными затратами, что недопустимо.

Для ориентированного взвешенного графа разработана программная реализация решения задачи о нахождении кратчайшего пути между парой вершин с учетом дополнительных ограничений в виде множества запрещенных вершин и дуг.

Несложно видеть, что на выходе можно узнать, сколько реально найдено путей минимальной длины через заданную вершину, если подсчитать количество путей, ведущих из стартовой вершины в заданную и из заданной в финишную вершину. Также подсчитывается, сколько прошло путей минимальной длины через заданную дугу. Данный подход используется в задаче маршрутизации в чрезвычайных условиях (глава 4).

Задачи маршрутизации на графах небольшой размерности можно решать с помощью точных классических алгоритмов (минимальный обзор приведен). В прикладной теории графов, предназначенной для решения многообразных задач прокладки маршрутов (замкнутых, разомкнутых, кратчайших, критических и т. п.) в сложных сетях, возникает ряд ограничений, условий, предписаний: как в стационарном случае, когда задача решается заранее и при всех известных условиях, так и нестационарном, когда информация появляется в процессе прокладки маршрута мобильным объектом (интеллектуальным агентом). Ограничения могут зависеть от времени, в виде фрагмента сети, доступной для анализа. Эта часть является перспективной как для дальнейших исследований по применению эвристических алгоритмов, так и для разработки программного наполнения системы по управлению локальными агентами, которые прокладывают оптимальные маршруты, обеспечивающие достижение гло-

бальной цели. Представляет интерес случай многократного повторения поиска в частично изменяемой сложной сети. Соответствующие задачи рассмотрены в 4 главе.

3.2 Метаэвристические алгоритмы в задачах многоагентной маршрутизации

Задача $mTSP$ формулируется как обобщение задачи TSP для многих коммивояжеров.

Разнообразие публикаций по этой тематике подтверждает необходимость разработки приближенных алгоритмов выбора оптимальных маршрутов в сложных сетях — на графах большого размера.

В статье автора [18] показано, что для $mTSP$ учет дополнительной информации меняет математическую постановку задачи и алгоритмы ее решения. В многоагентном подходе возникает свой класс задач, обусловленных поведением и управлением интеллектуальными агентами в условиях взаимодействия, например, в анализе алгоритмов маршрутизации, основанных на кластеризации графов. В работе [36] предполагается, что сочетание различных постановок задач (псевдобулевой оптимизации большой размерности) с кластеризацией (декомпозицией), с применением генетических и эволюционных алгоритмов (метаэвристик [136]) позволяет эффективно применять многоагентный подход для решения задач $mTSP$.

3.2.1 Гибридные алгоритмы многоагентной маршрутизации

Решение $mTSP$ с помощью сведения к TSP или релаксации получено в работах Bellmore и Hong [156] и Bektas [155]. Точные подходы к решению $mTSP$ основаны либо на преобразовании $mTSP$ в TSP, либо на ослаблении некоторых ограничений проблемы Bellmore, Hong [156] и Bektas [155]. Первая попытка решить $mTSP$ без преобразования в стандартный TSP предложена Laporte и Nobert [214], которые предложили два точных алгоритма: прямой и обратный. Прямой алгоритм основан на начальном ослаблении ограничений *исключения субтура* (SEC). Если обнаружено нарушение, то в задачу вводится ограничение, связанное с каждым субтуром, чтобы удалить нарушающий субтур. Обратный алгоритм также ослабляет SEC, но отличается проверкой целостности до того, как будет найдено решение.

Ali и Kennington [146] предложили алгоритм ветвей и границ, который использует лагранжевую релаксацию и субградиентный алгоритм для решения двойственной лагранжевой задачи, проблемы, которая моделируется, как асимметричный $mTSP$. Алгоритм может применяться как к симметричным, так и к асимметричным $mTSP$. Gromicho, Paix и Bronco [189] предложили другой точный алгоритм для асимметричных $mTSP$, который использует ослабление квази-присваивания (QA) определенных SEC в проблеме. Авторы также внедрили аддитивные ограничивающие процедуры для улучшения результатов нижней границы, для которых QA-релаксация работает плохо.

Gavish и Srikanth [183] разработали первый алгоритм для решения крупномасштабного $mTSP$. Они использовали метод ветвей и границ, устанавливая нижние границы с помощью лагранжевой релаксации. Авторы пришли к выводу, что их алгоритм работает лучше, чем алгоритмы, предложенные Laporte и Nobert [214]; Ali и Kennington [146].

Рассмотрим некоторые эвристические подходы. Bektas [155] рассматривает подходы к решению $mTSP$, основанные на преобразовании.

Самым распространенным для решения $mTSP$ является генетический алгоритм (GA). В работе [251] $mTSP$ предлагается в качестве модели глобальной задачи оптимизации сбора нектара цветов, например, нектарными летучими мышами. Модель включает в себя несколько независимых животных и много цветов, содержание нектара в которых зависит от времени. С помощью GA найдено оптимальное значение целевой функции с экспериментально полученными параметрами. Для определения расстояний применяется распределение Леви, что типично для естественных собирателей нектара. В отличие от многих моделей, в статье не делается предположений о природе распределения дальности полета. Представлены данные полевых экспериментов в Коста-Рике. Кроме подтверждения работоспособности модели, авторы обнаружили, что летучие мыши способны запоминать положения источников пищи и частично оптимизировать свои маршруты.

В статье [255] рассматривается $mTSP$ с минимизацией суммы расстояний всех маршрутов. Модель включает несколько складов, закрытый путь и требование минимального числа городов, которые должен посетить каждый агент. Для решения задач предлагается два GA. Первый применяется в сочетании с выбором рулетки и элитарным выбором, в котором предлагаются четыре новых вида мутационной операции. Второй связывает отбор и мутацию вместе. Применяются новый оператор селекции и более полный оператор мутации. Для сравнительного анализа приводятся *алгоритм оптимизации роя частиц* и *алгоритм оптимизации инвазивных сорняков*. Алгоритмы проверяются с помощью общедоступных тестов *TSPLIB*. Производительность оценивается с помощью серии сравнительных экспериментов. Показано, что второй GA дает лучшие результаты в сравнении с алгоритмами роя частиц и инвазивных сорняков.

В работе [184] предложены новые операторы для основных шагов GA: скрещивания и инициализации популяции. Основанная на теории групп методика генерации обеспечивает уникальность членов в популяции, и следовательно, отсутствие избыточности в пространстве поиска, а также устраняет эффект случайной инициализации. В предложенном операторе скрещивания расстояние Хэмминга сохраняется и существует очень мало шансов произвести нового потомка, который совпадет с членом популяции. Для эффективного представления пространства поиска применяется метод представления нескольких хромосом для кодирования пространства поиска $mTSP$. Авторы оценивают и сравнивают предложенную методику с методами, включающими описанные в статье [250] операторы скрещивания для двух стандартных целевых функций. Экспериментальные результаты показывают, что предложенный GA дает лучший результат.

В статье [138] предлагается новый эффективный GA с локальными операторами для решения $mTSP$ в разумные сроки для реальных приложений. Два новых локальных опе-

ратора предназначены для ускорения конвергенции процесса поиска и повышения качества решения. Результаты показывают, что алгоритм находит лучший набор путей с экономией 9,62% в среднем по стоимости.

Lixin, Jiyin, Aiying и Zihou [140] предложили модифицированный генетический алгоритм (MGA) для планирования горячей прокатки стали и пришли к выводу, что MGA дает почти оптимальные решения и показал улучшение на 20 % по сравнению с ручной системой.

Carter и Ragsdale [166] использовали метод, основанный на GA, для решения $mTSP$. Авторы провели исследование характеристик существующих представлений хромосом и предложили новое представление хромосомы из двух частей. Пространство поиска меньше, чем у предыдущих представлений хромосом. Кроме того, двухкомпонентное представление хромосом лучше, поскольку количество агентов увеличивалось (один из положительных показателей), особенно когда индивидуальные туры агентов должны быть сокращены для того, чтобы сбалансировать их продолжительность.

Király и Aboni [207] предложили мультихромосомный GA, выделяя по одной хромосоме для каждого агента. Были реализованы новые операторы мутации, такие как внутримаршрутные и перекрестные мутации. Авторы обнаружили, что мультихромосомный метод сходится к оптимуму быстрее, чем двухчастный метод, и сокращает общее пространство поиска проблемы. Они утверждали, что этот подход более эффективен, чем любой другой подход GA на сегодняшний день.

Blum [161] объяснил, как алгоритм муравьиной колонии (ACO) можно применить к непрерывной оптимизации, и обсудил новую тенденцию в ACO, которая использует метод гибридной оптимизации для решения $mTSP$. Junjie и Dingwei [201] попытались решить ограниченную версию $mTSP$, которая имеет верхнюю границу максимального количества городов для посещения агентом. Решение для $mTSP$ разработано с использованием алгоритма ACO, и его производительность проверена с помощью тестов из $TSPLIB$. Исследователи обнаружили, что MGA работает немного лучше, чем их алгоритм для небольших наборов данных, но их алгоритм превосходит алгоритм MGA для больших наборов данных.

Новый алгоритм ACO $NMACO$ для решения $mTSP$ был представлен в Majid и др. [220]. $NMACO$ улучшил характеристики классического ACO за счет выхода из точек локальных минимумов. Кроме того, по сравнению с алгоритмом *Sweep Algorithm + Elite Ant System* и *Modified Ant Colony Algorithm*, $NMACO$ показал лучшие результаты в пяти из шести тестовых случаев.

Krishna и др. [212] предложили другой подход ACO, который можно применить для решения $mTSP$ с ограничениями возможностей. Авторы избежали застоя и преждевременной конвергенции, используя стратегию распределения исходных муравьев и динамическое эвристическое обновление параметров на основе энтропии. Результаты экспериментов и сравнение производительности показали, что предложенный подход дает лучшие результаты, чем алгоритм ACO.

Aditi, Manas и Manoranjan [144] сформулировали некоторые TSP как линейные программы с неточными данными. Стоимость и время или и то, и другое минимизируются с

помощью гибридного эвристического алгоритма, сочетающего *ACO* и *GA*. Авторы разработали алгоритм, который позволяет решать одно- и многоцелевые ограниченные большие *TSP* с четкими, нечеткими и грубыми данными. Исследователи утверждают, что их алгоритм превосходит другие негибридные алгоритмы и может использоваться для различных сценариев реальной жизни.

Также в литературе встречается использование гибридных алгоритмов на базе *GA*. В [141] предлагается новый гибридный подход, который представляет собой комбинацию трех алгоритмов: *MACO* (модифицированной колонии муравьев), 2-*Opt* и *GA*. С помощью *ACO* генерируются решения, на которых применяется алгоритм 2-*Opt* для их улучшения. Далее происходит улучшение качества решений посредством *GA*. Причина объединения вышеупомянутых алгоритмов заключается в использовании их сильных сторон как в глобальном, так и в локальном поиске. Предлагаемый подход оценивается с участием различных экземпляров данных из стандартных контрольных показателей. Применяя критерии *TSPLIB* для больших задач, предложенный алгоритм показывает лучшие результаты, чем нынешний наиболее известный модифицированный алгоритм локального поиска гравитационной эмуляции (*M-GELS*) подход [150]. Для задач меньшей размерности он демонстрирует лучшие результаты, чем другие подходы и сопоставимые результаты с алгоритмом *M-GELS*.

Sedighpour и др. [236] предложили гибридный метаэвристический алгоритм локального поиска гравитационной эмуляции (*GA2OPT*), состоящий из *MGA* и локального поиска 2-*Opt*. На каждой итерации авторы сначала применяли модифицированный генетический алгоритм для получения решения, а затем использовали локальный поиск 2-*Opt* для улучшения результатов и повышения производительности алгоритма. *GA2OPT* сравнивался с различными метаэвристическими алгоритмами, такими как *MGA* и *MACO*. Результаты показали, что *GA2OPT* может найти лучшие решения, чем *MGA* и *MACO* в некоторых случаях и хуже в других случаях.

Sho, Haruna и Yoshifumi [241] предложили гибридный алгоритм *GIACO*, состоящий из генетического алгоритма и интеллектуальной и тупой оптимизации колоний муравьев (*IDACO*). Интеллектуальные муравьи следуют за феромоном и используют генетическую информацию, тогда как тупые муравьи в результате мутации *GA* не могут следовать за феромоном или использовать генетическую информацию. Авторы утверждают, что *IDACO* с тупыми муравьями дал лучшие результаты, чем обычный *ACO*, который состоит только из умных муравьев. *GIACO* сравнивали с *ACO*, *GA* и *GA-ACO*, состоящей только из умных муравьев. Авторы утверждали, что *GIACO* получил лучшие результаты, чем *GA-ACO*, который сам показал лучшие результаты, чем традиционные алгоритмы *ACO* и *GA*.

Shokouhi, Farahnaz, Hengameh и Hosseinabadi [242] предложили *M-GELS* для решения симметричного *mTSP*. Используя этот подход, алгоритм развертки используется для генерации набора возможных решений, которые будут улучшены с помощью *M-GELS*. Авторы пришли к выводу, что *M-GELS* превосходит известные алгоритмы оптимизации для решения *mTSP*.

Yousefikhoshbakht и Sedighpour [253] предложили *SW + ASelite* — комбинация алгоритма развертки, алгоритма элитной колонии муравьев и 3-*Opt* локального поиска. Авторы

использовали алгоритм поиска для генерации набора возможных решений, позже улучшенных с помощью элитной системы муравьев и 3-*Opt* локального поиска. Они заявили, что сила этого алгоритма в том, что для получения хорошего решения требуется минимальное время и несколько итераций. *SW + ASelite* сравнивался с алгоритмами *MGA* и *MACO* и дает лучшие решения для крупномасштабных задач по сравнению с *MGA*. По сравнению с *MACO SW + ASelite* превосходил во всех случаях, кроме pr226.

Разделение задачи между роем агентов не только позволяет сократить время поиска, увеличить вероятность нахождения глобального лучшего решения, но и создает широкий простор для применения многоагентных алгоритмов от оптимизации перевозки грузов до роя боевых роботов. Введение нескольких роев возникает в случае преобразования $mTSP$ в несколько обычных задач коммивояжера. Область применения такого подхода только расширяется, поскольку зачастую при решении вопроса перевозки грузов количество агентов-перевозчиков больше одного, и тогда необходимо разделить, например, города между агентами, как можно более эффективно [196].

В [148] осуществляется группирование городов в кластеры, где каждый кластер представляет собой набор смежных городов, а затем используется один из хорошо известных оптимизационных подходов для поиска оптимального маршрута для каждого кластера. Авторы применяют алгоритм муравьиных колоний, а также *GA* для последовательного и параллельного программирования.

Для решения $mTSP$ в [250] ученые рассмотрели две различные целевые функции. Задача первой — свести к минимуму общее расстояние, пройденное всеми коммивояжерами, а второй — свести к минимуму максимальное расстояние, пройденное любым коммивояжером. Вторая целевая функция касается баланса рабочей нагрузки между коммивояжерами. Также предложены два метаэвристических подхода к $mTSP$. Первый подход основан на *алгоритме искусственной пчелиной колонии*, а второй — на алгоритме оптимизации инвазивных сорняков. Применен локальный поиск для дальнейшего улучшения решения, полученного с помощью представленных подходов. Вычислительные эксперименты показывают превосходство предложенных алгоритмов над всеми другими современными подходами к этой задаче для обеих целевых функций.

В [224] авторами предложена система муравьиных колоний. Описаны два алгоритма, сочетающие кластеризацию K-средних и нечетких C-средних с системами муравьиных колоний. Экспериментально исследуется эффективность предложенных алгоритмов с целевой функцией, рассчитывающий общую длину/стоимость решения и степень его балансировки, измеряемую как амплитуда его подуровней.

Алгоритм *AC2OptGA* [141] преобразует найденное решение TSP в $mTSP$ решение. Тестировался на графах большой размерности $Pr439$ и $Pr1002$ (439 и 1002 вершин соответственно). Такой подход эффективен, если за приемлемое время решается TSP. Для очень большой размерности необходимы другие подходы (кластеризация, декомпозиция и др.).

Работы [141; 148; 196] наиболее близки к подходам, применяемым в настоящей работе.

Таким образом, существует множество подходов к решению многоагентных задач маршрутизации. Публикации подтверждают целесообразность выбора алгоритмов, используемых в данной работе.

3.2.2 Алгоритмы, используемые в программных реализациях

Рассматриваются прикладные модели, методы и алгоритмы маршрутизации многих агентов-коммивояжеров в сложных сетях различной природы. Выбор алгоритмов, их обоснование и реализация определяются целями прикладных задач, вероятностными и метрическими характеристиками сетей, сложностью структуры соответствующих графов. Поэтому соответствующие методы и алгоритмы должны базироваться на методах локального поиска и основанных на них метаэвристиках [136]. Численная реализация и тестирование на различных прикладных задачах позволяют сформировать пакет прикладных программ маршрутизации в сложных сетях, что является актуальной задачей.

Утверждение 3.1. *Методология разработки алгоритмов решения задач маршрутизации может быть основана на формировании по исходной сложной сети более простой (относительно реализации алгоритмов маршрутизации) по своей структуре сети.*

Ниже предлагается обобщенная многоагентная реализация алгоритма, который ранее не был конкретизирован для многоагентного подхода.

Для ряда задач *ДО* на графах существуют хорошие алгоритмы в случае планарных, метрических графов с весовыми коэффициентами, удовлетворяющими неравенству треугольника $l_{ij} \leqslant l_{ik} + l_{kj}$, $l_{ij} \geqslant 0$ (см. также пункт 1.1.1). В реальных задачах между вершинами заданы расстояния $c_{ij} \geqslant 0$, где $(i,j) \in U$ — множество дуг, $i \in V$ — множество вершин. Зная координаты вершин $i \in V, |V| = n$, можно поставить расстояниям c_{ij}, не удовлетворяющим неравенству треугольника, расстояния $l_{ij} = \rho(i,j)$, равные расстоянию по прямой между вершинами i и j. Такого типа задачи возникают для *задач облета вершин* (объектов) на некоторой высоте над поверхностью. Далее используем алгоритм 3.1 с наполнением, соответствующим дополнительным знаниям по $mTSP$.

Предложенный обобщенный алгоритм 3.1 инспирирован рядом актуальных прикладных задач: задачей планирования многодневных туристических маршрутов на инфраструктурной сети достопримечательностей Крыма и задачей доставки ресурсов (воды) экипажами (агентами-коммивояжерами) по территории Большой Ялты в условиях *ЧС*. *МАС* маршрутизации в условиях *ЧС* является достаточно сложной для моделирования системой. Для такой *МАС* необходим набор протестированных приближенных эвристических алгоритмов, композиция которых решает поставленную задачу.

Рассмотрим более подробно используемые в работе (наиболее известные) приближенные алгоритмы для решения задач маршрутизации: алгоритм ACO, алгоритм пчелиного роя, метод имитации отжига. Алгоритм ACO [175] основан на поведении муравьев в муравьиной

колонии, которая является многоагентной системой, где каждая отдельная особь или агент выполняет набор простых операций и в одиночку не способен обеспечить глобальное решение поставленной задачи. Сложность данного алгоритма зависит от времени жизни колонии t_{max}, количества вершин n и количества муравьев в колонии m.

Алгоритм 3.6. *Классический алгоритм ACO*

1: *Инициализируем граф* G (*например, вводим матрицу расстояний* D).
2: *Задаем параметры* α, β, Q, $t_{\max}$, m.
3: *Задаем значение маркера для каждого ребра* $\delta_{ij} \in G$ (*феромон*).
4: *Помещаем муравьев в произвольно выбранные вершины, без повторов.*
5: *Задаем произвольный кратчайший маршрут* T^* *и его длину* L^*.
6: *Создаем цикл по времени жизни колонии от* $t = 1$ *до* $t_{\max}$.

6.1: *Создаем цикл обхода всех муравьев колонии* $k = 1$ *до* $k = m$. *Строим путь* $T_k(t)$ *и рассчитываем длину* $L_k(t)$. *Вероятность перехода муравья* k *в город* i *из города* j *определяется формулой*

$$P_{ij,k} = \begin{cases} \tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t) \,/ \sum\limits_{j \in J_{ik}} \tau_{ij}^{\alpha}(t)\,\eta_{ij}^{\beta}(t), \\ 0, \end{cases}$$

где $j \in J_{ik}$,α *и* β *задают значимость уровня феромона и видимости города при выборе следующего города. При* $\alpha = 0$ *будет выбран ближайший город, что соответствует жадному алгоритму. Если* $\beta = 0$, *тогда работает лишь феромонное усиление, что влечет за собой быстрое вырождение маршрутов к одному субоптимальному решению.*

6.2: *Конец цикла обхода всех муравьев.*
6.3: *Проверка каждого полученного* $L_k(t)$ *на лучший маршрут по сравнению с* L^*.
6.4: *Если* $L_k(t)$ *лучше, чем* L^*, *обновить* L^* *и* T^*.
6.5: *Создаем цикл обхода всех ребер графа. Обновляем* $\tau_{i,j}$ *согласно правилам*:

$$\Delta\tau_{ij,k}(t) = \begin{cases} \dfrac{Q}{L_k(t)}, (i,j) \in T_k(t), \\ 0; \end{cases}$$

где τ_{ij} — *количество феромона на ребре* (i,j); $\eta_{i,j} = 1/D_{ij}$, *где* D_{ij} — *длина ребра* (i,j), J_{ik} — *список городов, которые необходимо посетить муравью* k, *находящемуся в вершине* i. *Здесь* $T_k(t)$ — *маршрут, пройденный муравьем* k *на итерации* t, $L_k(t)$ — *длина этого маршрута,* Q — *некоторый регулируемый параметр. Правило обновления феромона:*

$$\tau_{ij,k}(t+1) = (1 - p)\,\tau_{ik}(t) + \Delta\tau_{ij}(t),$$

где $\Delta\tau_{ij}(t) = \sum\limits_{k=1}^{m} \Delta\tau_{ij,k}(t)$, m — *количество муравьев,* p — *регулируемый параметр испарения феромона, принадлежащий отрезку* $[0;1]$.

6.6: *Конец цикла по ребрам.*

7: *Конец цикла по времени жизни колонии.*

8: *Передать на вывод кратчайший маршрут* T^* *и его длину* L^*.

Для рассмотрения алгоритма пчелиного роя [82; 204] введем следующие обозначения: s — число пчел-разведчиков; m — число выбранных точек (решений) для дальнейшего исследования $(m < s)$; e — число лучших (элитных) точек $(e < m)$; s_e — число пчел для более полного исследования e элитных решений; s_m — число пчел для более полного исследования оставшихся $(m - e)$ выбранных решений; Δ — размер окрестности, в которой пчелы выполняют более тщательный поиск.

Алгоритм 3.7. *Алгоритм пчелиного роя (PSO)*

1: *Случайно формируется* s *решений, каждое из которых представляет собой одну пчелу-разведчика.*

2: *Выбрать лучшие* m *из этих решений и провести локальный поиск. Для этого в их* Δ*-окрестности рассматриваются* r *случайных точек* ($r = s_e$ *или* $r = s_m$ *в зависимости от элитности данной точки), из которых отбирается лучшая.*

3: *Оставшиеся* $(s - m)$ *решений отбрасываются и заменяются на случайные точки из пространства решений.*

4: *Алгоритм останавливается, когда выполняется некое условие останова (достигнута необходимая точность, исчерпано число итераций и т. п.).*

Метод имитации отжига [208] основан на идее поведения материального тела при отвердевании, когда применяется процедура отжига, во время которой температура последовательно понижается до нуля. В ходе «отжига» металл сначала нагревают до некоторой температуры, из-за чего атомы покидают позиции в кристаллической решетке. Затем начинается медленное и контролируемое охлаждение. Атомы стремятся попасть в состояние с меньшей энергией, однако с определенной вероятностью они могут перейти и в состояние с большей. Эта вероятность уменьшается вместе с температурой. Переход в худшее состояние помогает отыскать состояние с энергией меньшей, чем начальная. Процесс завершается, когда температура падает до заранее заданного значения. В задаче *ДО* минимизация энергии представляется как целевая функция, где S — множество всех состояний — решений задачи, s_i — состояние на i-том шаге, t_i — температура на i-том шаге.

Алгоритм 3.8. *Алгоритм имитаций отжига*

1: *Инициализируем граф* G *(например, вводим матрицу расстояний* D*).*

2: *Задаем параметры* T_s *и* T_f.

3: *Задаем произвольное начальное состояние* S *и считаем энергию для этого состояния* E *по формуле*

$$E = \sum_{i \in S, i \neq n} w\,(i,i+1) + w\,(n,0)\,,$$

где $w\,(i,j)$ — *расстояние от вершины* i *до вершины* j.

4: *Создаем лучшее состояние* S^* *с энергией* E^*.

5: *Инициализируем цикл «остывания» до тех пор, пока* $T > T_f$.

5.1: *Случайным образом генерируем новое состояние-кандидат CS, считаем его энергию CE.*

5.2: *Сравниваем CE с E.*

5.2.1: *Если $CE > E$, то переходим в состояние-кандидат и делаем его текущим, т. е. $E = CE$; $S = CS$.*

5.2.2: *Если $CE < E$, то переходим в это состояние с вероятностью P, вычисленной по формуле $P_i = \exp(-\Delta E/T_i)$, где $\Delta E = CE - E$.*

5.3: *Сравниваем энергию текущего состояния E с энергией лучшего состояния E^*. Если $E < E^*$, то заменяем лучшее состояние на текущее, т. е. $E^* = E$; $S^* = S$.*

5.4: *Изменяем T по правилу $T_{i+1} = 0.1T_i/i$, где $T_0 = T_s$.*

6: *Конец цикла «остывания».*

7: *Выводим значение лучшей энергии E^* и соответствующего состояния S^*.*

В алгоритме приняты следующие обозначения: T_s — начальная температура; T_f — конечная температура; предполагается, что граф G полносвязный.

Генетический алгоритм основан на идее естественного отбора. В общем виде он формулируется следующим образом.

Алгоритм 3.9. *Схема генетического алгоритма GA*

1: *Создание начальной популяции.*

2: *Скрещивание и/или мутация.*

3: *Отбор.*

4: *Формирование нового поколения.*

4.1: *Если результат не достигнут — переход к пункту* 2.

4.2: *В противном случае текущая популяция является результирующей.*

Рассмотрим *модификацию генетического алгоритма* (MGA) для решения многоагентной задачи коммивояжера, в котором исходный граф разбивается на m кластеров и для каждого кластера решается обычная задача коммивояжера. Предложенный подход включает создание популяции из хромосом длины $n + m$, где n вершин представлены в виде перестановки чисел от 1 до n. Перестановка поделена на m подмаршрутов при помощи добавления m положительных чисел (от 1 до m), для отличия одного коммивояжера от другого. Пусть m_i отвечает числу вершин, пройденных i-м коммивояжером, тогда, согласно условию задачи, $K \leqslant m_i \leqslant L$.

Один из лучших кроссоверов с точки зрения скорости и качества — это двухточечный кроссовер. Предлагаемая модификация кроссовера состоит в том, что случайно выбранные точки разделяют родительские строки на левые и правые подстроки и выбираются родительские права — правила выбора подстрок. Далее процесс такой же, как и для двухточечного кроссовера, отличие состоит в том, что вместо выбора случайных нескольких позиций в родительском маршруте выбраны все позиции справа от случайно выбранной точки пересечения. В данном алгоритме может быть использован один из двух видов мутации: оператор выбирает две случайные точки (разрезающие точки) в строке и меняет их местами; возможность реверсирования всех точек, которые находятся между разрезающими точками. Данный

подход обуславливает нахождение лучшего решения, по сравнению с использованием генетического, «жадного» или алгоритма муравьиной колонии, примененных для решения обычной задачи коммивояжера для каждого кластера графа, как следует из [166]. В [131] показано, что снижение размерности задачи маршрутизации в прикладной теории сетей связывается с предобработкой доступных данных о сети, использованием особенностей взвешенных графов для более быстрого определения их характеристик (центр, радиус, диаметр и др.).

Утверждение 3.2. *Снижение сложности вычислений в построении рациональных приближенных решений задач $mTSP$ на сложных сетях большой размерности может быть достигнуто по следующей схеме*:

1) *на первом этапе решается задача распределения сети между коммивояжерами с помощью кластеризации* (*декомпозиции*);

2) *на втором этапе решаются задачи коммивояжера на каждом кластере с помощью метаэвристик*;

3) *в зависимости от полученного результата уточняются границы используемых кластеров*;

4) *в дальнейшем агенты-коммивояжеры могут обучаться и быть автономными.*

Подчеркнем, что в рамках интеллектуализированной MAC, которая ориентирована на кластеризацию задачи и дальнейшее распараллеливание, более точное значение дополнительной информации о сети и задаче позволит подключать композицию алгоритмов, наиболее подходящих для конкретного случая.

3.2.3 Синтез алгоритмов кластеризации для многоагентной задачи коммивояжера

Данный раздел базируется на работах автора [27; 28; 36]. При решении прикладных задач маршрутизации большое внимание уделено синтезу алгоритмов кластеризации для $mTSP$.

Рассматривается согласованная с основной задачей построения маршрутов в сложной сети задача кластеризации, которая позволяет синтезировать иерархические алгоритмы решения, существенно снижающие время построения приемлемого приближенного решения.

Любая сложная сеть — это граф большой размерности, состоящий из вершин и дуг, которым отвечают наборы количественных (метрических) и качественных характеристик, признаков. Может оказаться, что для некоторых частей сети (кластеров) выбранные свойства практически не меняются, но претерпевают изменения на границах кластера. В связи с этим цель кластеризации заключается в упрощении исходной сложной сети, выделении в ней характерных структур для последующего анализа кластеров и межкластерных связей, а также построения приближенных (рациональных, реального времени, быстрых) решений соответствующих задач $ДО$ на графах.

Результатом кластеризации является множество кластеров, которые полностью покрывают граф сети, либо границы кластеров, определяемые дугами разрезов между кластерами. В зависимости от исходных постановок задач *ДО* на сети возникает множество различных многоагентных задач на кластерах и межкластерных коммуникациях.

Исследование автором задач *ДО* на сложных сетях акцентируется на использовании дополнительной информации [18], алгоритмах взаимодействия интеллектуальных агентов в решении сетевых задач [35; 130], алгоритмах реоптимизации сети [25] и др. Далее в работе для решения задач маршрутизации применяются алгоритмы кластеризации, основанные на K-средних (K-$means$), бионических алгоритмах, и построении маршрутов с помощью генетических алгоритмов. Перспективной является композиция K-$means$, муравьиного и модифицированного генетического алгоритмов.

Для заданной (частично или полностью) сложной сети необходимо найти, согласованную с задачей маршрутизации, разметку сети на определенное количество кластеров, которая обеспечивает высокую точность и скорость решения соответствующих экстремальных задач на графах.

Подходящим способом для эффективного решения задач маршрутизации в сложноструктурированных сетях является использование математических моделей, описывающих коллективное поведение децентрализованной самоорганизующейся системы, состоящей из множества агентов, локально взаимодействующих между собой и с окружающей средой для достижения глобальной цели. В природе примерами подобного рода систем являются, например, муравьиные колонии, рои пчел. Каждый агент в системе функционирует автономно, используя простые правила. В то же время алгоритм совместного поведения всех агентов позволяет решить подобную задачу. Коллективный интеллект рассматривается как эффективная процедура оптимизации, которой присущи масштабируемость, возможность решать задачи независимо от их размерности, гибкость, отсутствие жесткой структуры, простота правил поведения агентов.

При переходе к математическим мультиагентным моделям коллективного интеллекта вводятся так называемые эвристические коэффициенты, которые являются управляющими параметрами мультиагентных методов и алгоритмов. От значения этих параметров зависит, насколько эффективно будет решена оптимизационная задача. Параметры могут принимать бесконечное число значений из некоторого диапазона. Поэтому встает вопрос об их подборе и необходимости проведения экспериментальных исследований на распространенных модельных (эталонных) тестовых задачах из библиотек, чтобы выяснить оптимальные значения коэффициентов и оценить вычислительную сложность методов коллективного интеллекта. Иными словами, необходимо выяснить, насколько точно такие методы могут решать задачи *ДО* по сравнению с известными методами; какова оценка их вычислительной сложности, а также каковы оптимальные значения используемых в методах эвристических коэффициентов, и как они влияют на конечный результат.

С учетом указанных подходов представим результаты по разработке алгоритмов кластеризации, пригодных для решения задачи многих коммивояжеров и их программной реализации.

Методы кластеризации условно можно разбить на автоматические и интерактивные, использующие априорную информацию, или универсальные, которые не используют априорную информацию (о компактности, однородности и др.). К универсальной группе методов и алгоритмов относят алгоритм $K\text{-}means$; алгоритмы, использующие метрические характеристики; алгоритмы построения различных разрезов и др. [16; 68; 81; 120].

Алгоритм $K\text{-}means$ предполагает быстрый кластерный анализ путем выделения k кластеров, которые располагаются на максимальном расстоянии друг от друга [215]. Число кластеров k выбирается интуитивно либо опирается на результаты экспериментов [246]. Идея алгоритма состоит в том, что центры кластеров соответствуют локальным максимумам плотности распределения данных. Базовый алгоритм $K\text{-}means$ предполагает случайный или эвристический выбор k центров кластеров, размещение каждой вершины графа в кластер с ближайшим центром к этой вершине, после чего заново пересчитываются центры кластеров до сходимости процесса. Алгоритм гарантированно сходится, но не обязательно приводит к оптимальному решению, поскольку зависит от начального множества кластеров и значения k. Вычислительная сложность алгоритма $K\text{-}means$ равна $O(nkt)$, где n — размерность сети, k — число кластеров, а t — число итераций, которые требуются алгоритму для приведения кластеров к стабильному состоянию. К недостаткам алгоритма относится его чувствительность к локальным изменениям сети (запреты, предписания, удаление элементов и т. п.), снижение скорости работы на больших объемах данных, необходимость предварительно указывать число кластеров k [193].

Метод разреза графа предполагает, что сложная сеть представляется в качестве взвешенного неориентированного графа. Между всеми вершинами строятся ребра, веса которых показывают меру сходства между вершинами по какой-либо характеристике. Исходная сеть в виде графа разрезается на подграфы, чтобы внутри подграфа веса ребер значительно отличались от весов ребер, связывающих подграфы. Одними из лучших графовых алгоритмов на сегодняшний день считаются алгоритм $NormalizedCut$ [239] и его модификации [181; 205], алгоритм $NestedCuts$ [249], а также алгоритм SWA [238; 246]. Большинство алгоритмов минимального разреза графа имеют трудоемкость $O(n)$, где n — общее число вершин графа, но требуют значительных затрат памяти для хранения матриц расстояний размером $n \times n$. Поэтому их затруднительно применять к сетям большой размерности. Таким образом, проблема построения алгоритмов кластеризации всей сети на основе кластеризации графов с приемлемыми требованиями к памяти и быстродействию остается актуальной.

Автоматические методы и алгоритмы не позволяют находить решение произвольных задач кластеризации с гарантированным результатом. Ни один из автоматических методов и алгоритмов не идеален, в лучшем случае необходимо использовать гибридную кластеризацию из разных комбинаций алгоритмов кластеризации, полученных разными методами с разными параметрами. Применительно к задачам кластеризации изображений как графовых структур в [142] было проведено сравнение различных бионических подходов. Рассмотрено 80 различных бионических методов. Сравнивались генетические алгоритмы (GA), эволюционные стратегии (ES), генетическое программирование (GP), метод роя частиц (PSO), метод

муравьиных колоний (ACO). В качестве критерия сравнения для алгоритмов кластеризации, как правило, используют точность работы и скорость выполнения.

Для алгоритма кластеризации важное значение имеет сложность обрабатываемой сети. Роевые методы кластеризации, в частности *Darwinian PSO* и *Fractional-Order Darwinian PSO*, описаны в [149; 222], особенностью применения которых является возможность нахождения квазиоптимальных решений, которые рекомендуется учитывать при выборе между скоростью работы и качеством решения.

Наилучшую точность кластеризации для сложноструктурированных сетей показывают муравьиные алгоритмы, в частности, для задач обработки изображений [178; 215; 232]. Данные этих исследований показали значительное влияние целевых функций муравьиных алгоритмов на процесс разметки сети на кластеры, что позволяет сделать вывод о зависимости точности результата от используемых критериев целевой функции. Анализ алгоритмов эволюционных стратегий [254] и генетических алгоритмов [228] показал, что данный подход можно применять для обработки сложно структурированных сетей, но результат обладает меньшей степенью точности, а сходимость наблюдается в меньшем проценте случаев.

В качестве задачи *ДО* возьмем $mTSP$ для $m = k$ агентов. Искомый граф будет иметь большое количество вершин, поэтому для решения $mTSP$ будет применяться один из эвристических алгоритмов. Граф имеет полную связность.

Решаются две задачи: нахождение оптимальных центров кластеров; формирование оптимальных кластеров на основании критерия качества и решение $mTSP$ на кластерах.

Нахождение оптимальных центров кластеров. В качестве исходных данных задаем количество кластеров k. Будем считать, что оптимальность центра кластера задается двумя параметрами: удаленность центра от других центров кластеров; выполнение «гипотезы компактности» для текущего кластера. *Гипотеза компактности* в задачах кластеризации — предположение о том, что схожие объекты гораздо чаще лежат в одном кластере, чем в разных или, другими словами, что кластеры образуют компактно локализованные подмножества в пространстве объектов. Это также означает, что граница между кластерами имеет достаточно простую форму.

Для решения этой задачи подойдет алгоритм кластеризации $K\text{-}means$. В классической реализации алгоритма $K\text{-}means$ не требуется задавать искомые центры, а достаточно знать количество кластеров. Следовательно, алгоритм способен сам подобрать оптимальные для него кластеры (подграфы). Разбив граф на соответствующие подграфы, можно взять центр каждого подграфа, где центр — множество вершин, имеющих минимальный эксцентриситет. Нам достаточно одной вершины из каждого такого множества. Соответствующие вершины и будут оптимальными центрами для исходного графа с заранее заданным количеством кластеров.

Предпочтение алгоритма $K\text{-}means$ можно аргументировать тем, что этот алгоритм показывает хорошие результаты кластеризации с различными критериями качества. Единственным альтернативным вариантом со схожим уровнем точности можно отметить вероятностный EM-алгоритм. Но он имеет несколько недостатков, в следствие которых получить

с его помощью необходимые центры не представляется возможным, хотя бы потому, что алгоритм не предназначен для «навязывания» ему количества кластеров.

Алгоритм *K-means* также придерживается гипотезы компактности (форма кластеров — сферическая), поскольку использует в качестве критерия «схожести» расстояние от вершины до центра кластера. Отсюда вытекает и тот факт, что центры кластеров будут максимально удаленными друг от друга.

Нахождение оптимальных кластеров на основании критерия качества. Рассмотрим *TSP* для k агентов. Имеем исходный граф, на котором выделены базы для каждого агента (см. раздел 1.2 и главу 2). Будем считать, что каждому достанется «приблизительно» равный подграф искомого графа. Ставится задача: по искомым центрам разбить граф на k кластеров $G_i, i = \overline{1,k}$ так, что все агенты проходят почти равные расстояния $f_i(G_i) \approx f_{среднее}, i = \overline{1,k}$. Равенство кластеров относительно *TSP* будем оценивать по следующему критерию качества:

$$\left| \max_{1 \leqslant i \leqslant k} f_i(G_i) - \min_{1 \leqslant i \leqslant k} f_i(G_i) \right| \leqslant \varepsilon \tag{3.1}$$

В качестве целевой функции выбрано значение пройденного пути в *TSP*. Нахождение кратчайшего пути будет осуществляться *GA* с элитной селекцией, частично соответствующим скрещиванием и с использованием жадной эвристики. Данная модификация предполагает, что одна из хромосом *GA* в начальной популяции будет формироваться жадным алгоритмом (ближайший сосед). Такая хромосома гарантирует локальный оптимум, как и любая жадная эвристика, и при этом способствует более быстрой сходимости *GA*. Проблема «застоя» в локальном оптимуме решается тем, что во время селекции половина хромосом из предыдущего поколения останутся в новом поколении. Поэтому «слабые» хромосомы могут внести свой дальнейший вклад в выходе из локального оптимума.

Получив начальное разбиение графа на k кластеров и значения целевой функции для каждого подграфа генетическим алгоритмом, оптимизируем кластеры путем балансирования значений целевой функции для каждого кластера. Для этого необходимо «перебросить» вершины из самого крупного по значению целевой функции кластера G_{max} в самый малый — G_{min}. Возьмем ближайшую вершину из G_{max} до G_{min} и присвоим ее кластеру G_{max}. При этом если расстояние между кластерами большое и/или между ними есть другие кластеры, то необходимо передать вершину из G_{max} в G_{min} «транзитом» через все кластеры, лежащие между данными:

$$G_{max} \longrightarrow G_1 \longrightarrow \ldots \longrightarrow G_m \longrightarrow G_{min},$$

где G_i — кластеры между G_{max} и G_{min}, $i = \overline{1,m}$.

Будем считать, что кластер G_2 лежит между G_1 и G_3, если:

$$\begin{cases} \rho(G_1,G_2) < \rho(G_1,G_3), \\ \rho(G_2,G_3) < \rho(G_1,G_3), \end{cases}$$

где $\rho(x,y)$ — расстояние от центра кластера x до центра кластера y. После первого «перебрасывания» вершин применим генетический алгоритм для нового разбиения. Такие

«перебрасывания» будем совершать до тех пор, пока не выполнится условие остановки (3.1). При этом ε может быть сколь угодно малым. Впрочем, брать слишком малые значения не стоит ввиду того, что алгоритм может подобрать кластеры с очень большим разбросом вершин, что не является приемлемым. В программной реализации используется динамическое увеличение ε (порога), если алгоритм не смог подобрать кластеры за некоторое количество итераций.

Данный алгоритм позволяет подобрать оптимальные подграфы с сохранением заданных центров для каждого кластера. Разбиение почти всегда будет неприемлемым, если обнаружатся центры, которые будут расположены близко друг к другу. В таком случае необходимо сначала найти оптимальные центры, а потом применить данный алгоритм.

В наиболее простом случае под синтезом алгоритмов кластеризации и решения *TSP* будем понимать использование любого алгоритма кластеризации для получения начального разбиения исходного графа на подграфы и любого алгоритма решения *TSP*. В общем случае выглядит следующим образом.

Алгоритм 3.10. *Синтез алгоритмов кластеризации и решения mTSP для $m = k$ агентов*

1: *Для исходного графа ввести нужное количество кластеров k и их центры.*
2: *Найти начальное разбиение графа на k кластеров с использованием текущих центров.*
3: *Если начальное разбиение на текущих центрах оказалось неприемлемым, то подобрать новые центры с помощью K-means.*
4: *Переопределить кластеры с помощью механизма «перебрасывания» вершин.*
5: *На каждом кластере найти решение локальной TSP.*
6: *Найти значение целевой функции для каждого кластера текущего разбиения.*
7: *Проверить условие сходимости и, если оно не выполнено, вернуться к пункту* 4.

В качестве алгоритма кластеризации графа для начального разбиения подойдет любой алгоритм кластеризации с четкой принадлежностью вершин. *TSP* также может решаться любым из известных алгоритмов. Данный синтез алгоритмов применим не только для *TSP*, поскольку целевой функцией может быть целевая функция любой задачи *ДО* на графах, для которой необходимо применить кластеризацию и для которой можно сформулировать условие сходимости алгоритма синтеза.

В программных реализациях применялся алгоритм баланса нагрузки (*LBA*): назначение вершин агентами с процедурой *AC2OptGA*.

Численная реализация приведенных алгоритмов, их модификаций приводятся в разделе 3.3. Кроме алгоритмов, непосредственно связанных с численным решением задачи маршрутизации типа $mTSP$, используются алгоритмы кластеризации, построения разрезов, реоптимизации, в которых учитывается информация о задаче $mTSP$, т. е. используется композиция алгоритмов (в зависимости от поступающей информации, в рамках выбранной формализации модели $mTSP$).

3.3 Численная реализация алгоритмов

3.3.1 Программная реализация алгоритмов синтеза кластеризации и многоагентной маршрутизации

Рассмотрим применение алгоритмов кластеризации графов для задачи k коммивояжеров. На основе теоретических результатов предыдущих разделов разработана программная реализация следующих алгоритмов кластеризации: иерархический алгоритм, $K\text{-}means$ и жадный алгоритм с различными модификациями. Реализован GA для решения TSP, а также синтез алгоритмов кластеризации и решения TSP с нахождением оптимальных центров. В частности, используется алгоритм развертки (см. раздел 3.1.1).

Основной задачей работы алгоритма является нахождение оптимальных кластеров или подграфов искомого графа, с учетом равномерного распределения маршрутов коммивояжеров по кластерам. В результате показано отличие между разбиением графа на кластеры (с сохранением искомых центров в этих кластерах) и нахождением оптимальных центров для дальнейшей кластеризации с ними. Для поиска оптимальных по отношению к задаче k коммивояжеров подграфов разработан синтез алгоритмов кластеризации и решения задач $mTSP$. Суть метода заключается в механизме «перебрасывания» вершин из более крупных кластеров в более мелкие до тех пор, пока не будет выполнено условие сходимости, целевая функция которого зависит от длины путей коммивояжеров. Предусмотрены различные варианты этого механизма, включая транзитную передачу вершин через кластеры, находящиеся между крупными и мелкими кластерами, для того чтобы результирующие кластеры не потеряли компактность и не создавали пересечений в дальнейшем поиске оптимального маршрута.

Основными объектами исследования при решении поставленной задачи являются сами алгоритмы, при помощи которых можно выполнять обработку данных. Поставленная задача заключается в исследовании эффективности алгоритмов решения $mTSP$. Главным критерием оценки является длина вычисляемого пути, но получаемые алгоритмы должны быть также эффективными с точки зрения времени работы.

Вычислительный эксперимент. Среда программирования: Microsoft Visual Studio 2015 (Windows Forms); язык программирования: $C\#$. Исходные данные: полный граф (100 вершин), количество кластеров k и их центры. Рассмотрим задачу многих агентов-коммивояжеров с «плохими» начальными условиями, т. е. центры каждого агента будут расположены близко друг к другу. Для примера пусть число агентов будет 5, а центрами — вершины 1, 2, 3, 4, 5. Реализуем кластеризацию графа алгоритмом $K\text{-}means$ и жадным алгоритмом с распараллеливанием по исходным данным (рис. 3.4, 3.5), сохраняя текущие центры (вершины, выделенные голубым цветом).

Получив данные разбиения, применим алгоритм GA для нахождения оптимального пути для каждого подграфа (рис. 3.6, 3.7).

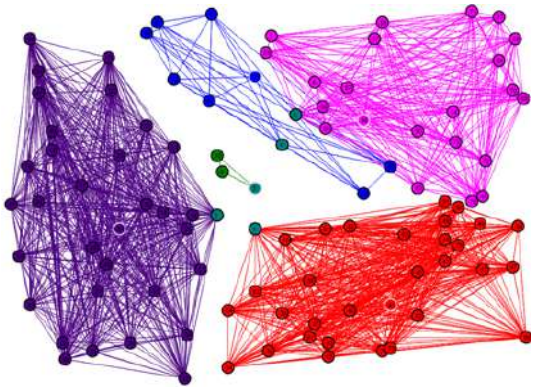


Рисунок 3.4 — Разбиение графа на пять подграфов: метод *K-means*

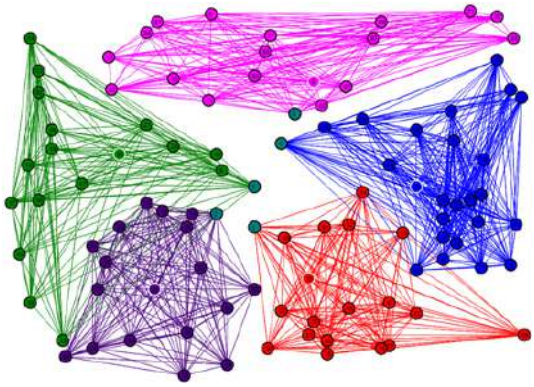


Рисунок 3.5 — Разбиение графа на пять подграфов: жадный алгоритм

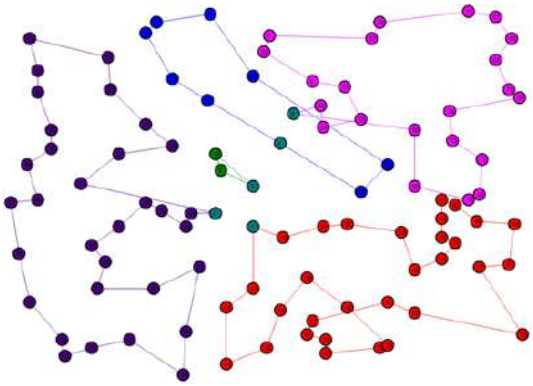


Рисунок 3.6 — Построение *GA* оптимального маршрута для кластеров, полученных: алгоритм *K-means*

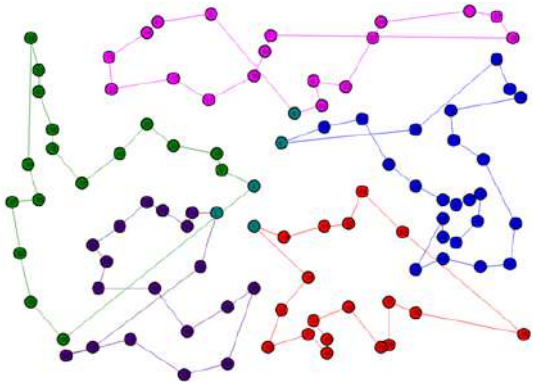


Рисунок 3.7 — Построение *GA* оптимального маршрута для кластеров, полученных: жадный алгоритм

Сравним результаты.

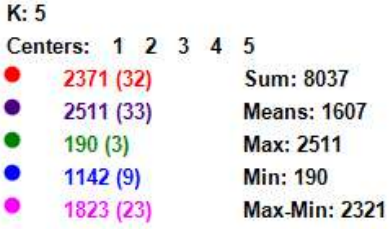


Рисунок 3.8 — Результаты работы: алгоритм *K-means*

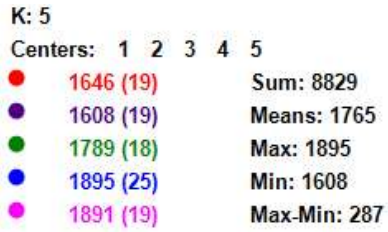


Рисунок 3.9 — Результаты работы: жадный алгоритм

На рис. 3.8, 3.9 слева отображены значения целевой функции для соответствующего кластера, а в скобках — количество вершин в подграфе; справа — дополнительная информация: сумма значений целевых функций, их среднее значение, минимальное и максимальное значение и разница между минимумом и максимумом.

Учитывая, что оптимальность кластеров характеризуется не только минимизацией значения целевой функции, но и ее усреднением по всем кластерам, можно однозначно сказать,

что жадный алгоритм с модификацией распараллеливания отработал лучше и при этом разница между наибольшим и наименьшим кластером невелика.

В общем случае жадный алгоритм с данной модификацией практически всегда будет давать приемлемый результат, и поэтому рекомендуем к использованию в качестве начального разбиения для синтеза алгоритмов.

Смысл модификации распараллеливания заключается в том, чтобы разбиение графа проходило по жадному алгоритму параллельно, когда кластеры ведут себя независимо друг от друга, за исключением того, что одна вершина не может одновременно принадлежать нескольким кластерам.

Найдем оптимальные центры для данного графа при $k = 5$ и снова применим те же алгоритмы кластеризации. Как видно из самих разбиений на рис. 3.10, 3.11 кластеры получились более компактными, с выдержанной формой (сферической) и без пересечений.

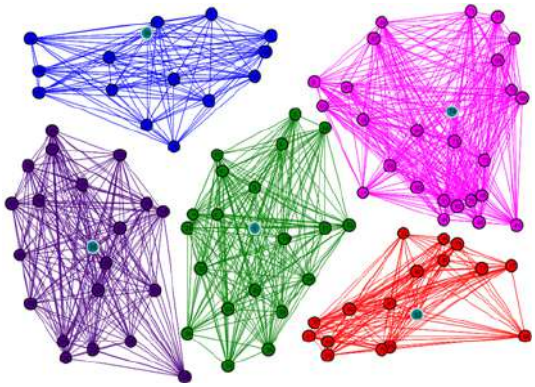


Рисунок 3.10 — Разбиение графа на подграфы с новыми центрами: метод *K-means*

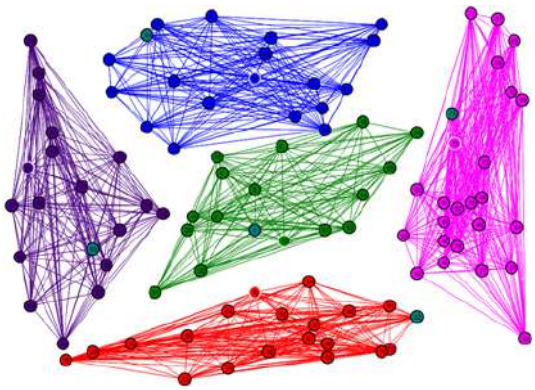


Рисунок 3.11 — Разбиение графа на подграфы с новыми центрами: жадный алгоритм

Применим *GA* для данных разбиений (рис. 3.12, 3.13) и сравним результаты. По данным на рис. 3.14, 3.15 видно, что для найденных центров для применения алгоритма *K-means*, теперь не наблюдается сильных расхождений между кластерами. Однако, жадный алгоритм отработал сравнительно одинаково как для искомых центров, так и для новых. Это еще раз подчеркивает, что жадный алгоритм с распараллеливанием дает адекватный результат.

Реализуем синтез алгоритмов кластеризации и решения *TSP* на кластере. В качестве начального разбиения используется *K-means* и жадный алгоритм с распараллеливанием; *GA* в качестве решения *TSP*.

Из графиков на рис. 3.16, 3.17 видно, что синтез приближает значения целевых функций для всех кластеров к среднему значению ($AvgSyn$), с некоторым отклонением ε — как один из параметров критерия качества (3.1). В данной реализации $\varepsilon = 100$, с последующим увеличением порога, если алгоритм долго не сходится. На рис. 3.20, 3.21 разница между максимальным и минимальным значениями целевой функции оказалась меньше $\varepsilon = 100$, а на рис. 3.18, 3.19 близко к ε. Это значит, что критерий качества (3.1) выполняется, но сходимость алгоритма сильно зависит от начального разбиения. По рис. 3.18, 3.19 можно наблюдать численную разницу между начальным разбиением и после реализации алгоритма синтеза.

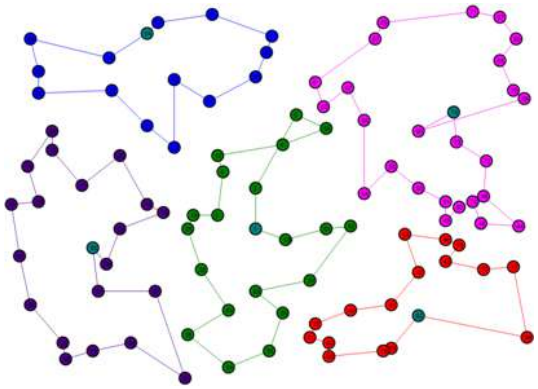


Рисунок 3.12 — Построение GA оптимального маршрута для кластеров с новыми центрами, полученных: алгоритм K-$means$

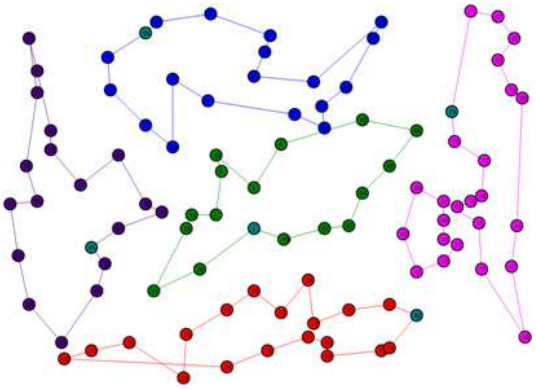


Рисунок 3.13 — Построение GA оптимального маршрута для кластеров с новыми центрами, полученных: жадный алгоритм

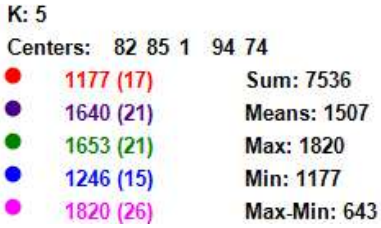


Рисунок 3.14 — Результаты: алгоритм K-$means$ с новыми центрами

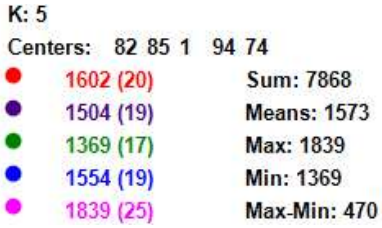


Рисунок 3.15 — Результаты: жадный алгоритм с новыми центрами

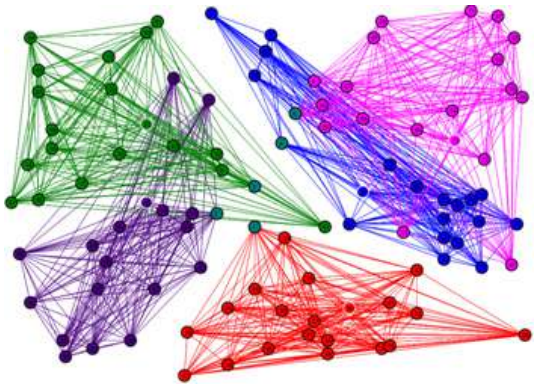


Рисунок 3.16 — Результат кластеризации синтезом: K-$means$ в качестве начального разбиения

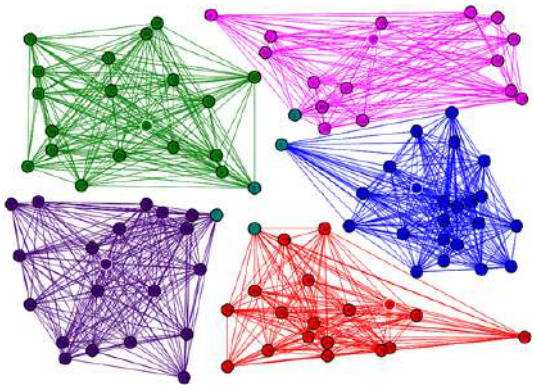


Рисунок 3.17 — Результат кластеризации синтезом: жадным алгоритмом в качестве начального разбиения

Кратко рассмотрим эффективность приведенных в главе 3 алгоритмов на графе $eil51$ [247] с 51 вершиной. Приведены средние за 10 опытов результаты для одного коммивояжера [129].

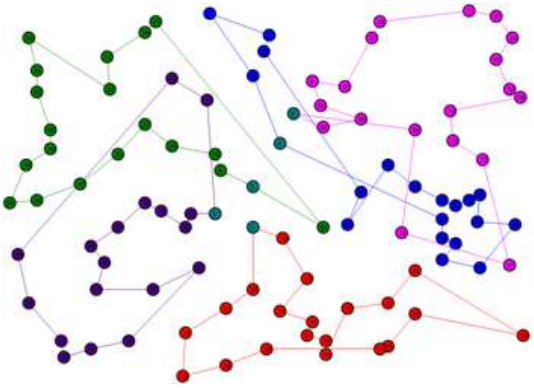


Рисунок 3.18 — Результат работы *GA* для подграфов, полученных с помощью синтеза: *K-means* в качестве начального разбиения

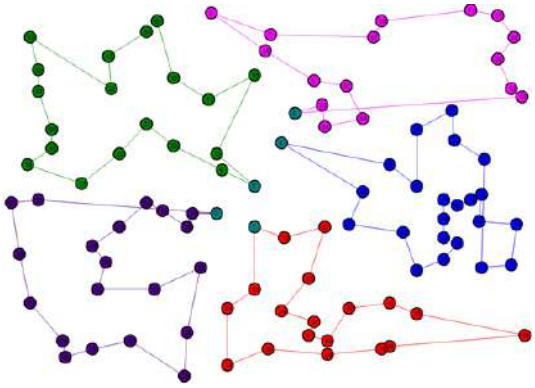


Рисунок 3.19 — Результат работы *GA* для подграфов, полученных с помощью синтеза: жадным алгоритмом в качестве начального разбиения

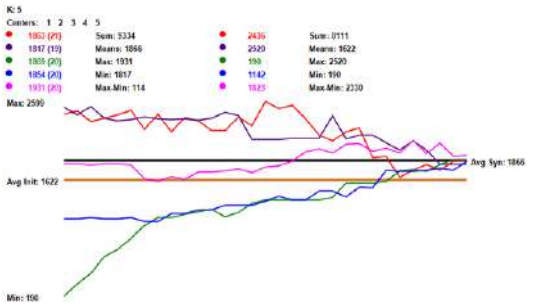


Рисунок 3.20 — Результаты синтеза с *K-means* в качестве начального разбиения

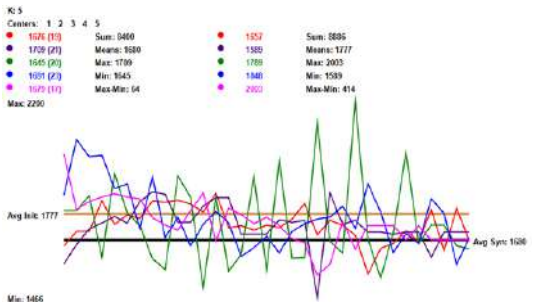


Рисунок 3.21 — Результаты синтеза с жадным алгоритмом в качестве начального разбиения

	оптимум	муравьиный	пчелиный	рой частиц	*GA*	*GA* мод.
счет:	426	1294.0	1363.7	852.5	968.9	985.9
время:	–	70.152	22.348	15.349	40.766	48.11

Оценим поведение алгоритмов на графе *bier*127 [247] с 127 вершинами.

	оптимум	муравьиный	пчелиный	рой частиц	*GA*	*GA* мод.
счет:	118282	393942	567855.2	386457	436889	433728
время:	—	417.4	55.1	375.3	170.9	186.4

Как видно, стандартные, не оптимизированные под задачу алгоритмы не приближаются к оптимальному результату, тем не менее, серьезно теряют в скорости работы при увеличении размера графа.

Пользуясь достаточно большим числом агентов, муравьиный (150 агентов, 80 итераций) и алгоритм роя частиц (450 агентов, 1200 итераций), хоть и показывают несколько лучший результат в сравнении с пчелиным и *GA*, но работают гораздо медленнее. В тоже время пчелиный и *GA* показывают лучший результат по времени, но худший по длине маршрута

при достаточно большом числе агентов (пчелиный — 550 агентов и 820 итераций, GA — 150 агентов, 1000 итераций). Случайность начальной выборки в пчелином и GA алгоритмах не позволяют сузить область поиска и обнаружить даже локальный минимум в сколь-нибудь короткие сроки, ввиду огромного числа решений.

Используем многоагентный (MA) жадный алгоритм в качестве начальной выборки для пчелиного и GA. Используются именно эти алгоритмы в виду адекватности их скорости работы. Результаты средние за 10 тестов.

	оптимум	пчелиный + MA жадный	GA + MA жадный
счет:	118282	130982.7	134458.9
время:	—	29.156	11.786

Оптимизация начальной выборки позволяет значительно сократить затраты памяти и ресурсов на увеличение размеров популяции пчел/особей и количества итераций. Как следствие, работа алгоритмов ускорилась при значительном улучшении качества результата.

Разделим граф на две части с помощью нахождения максимального разреза, увеличив число коммивояжеров до 2. Результаты средние за 10 тестов.

	оптимум	пчелиный + MA жадный	GA + MA жадный
счет1:	118282	103741.4	104098.2
время1:	—	14.969	3.5
счет2:	—	91881.3	92243
время2:	—	14.8	3.528
сумма:	—	195622.7	196341.2

В [223] содержится база тестовых примеров для задачи коммивояжера. Каждый блок тестовых данных содержит в себе список двумерных координат. С помощью этого списка можно построить решение задачи коммивояжера своими методами и сравнить с оптимальным решением, которое также входит в состав тестового примера.

На первом этапе было выбрано несколько тестовых примеров графов небольшой размерности, соответствующих кластеру сложной сети и проведен численный эксперимент с помощью реализованных алгоритмов. Результаты приведены в табл. 3.1 [91].

Программа предусматривает наглядную демонстрацию работы алгоритмов, в которой для сравнения приводится изображение неоптимального произвольного маршрута на графе и отображение маршрута, найденного в процессе работы конкретного алгоритма. Например, на рис. 3.22–3.24 показаны результаты применения алгоритма имитации отжига (одного из представленных в табл. 3.1).

Таблица 3.1 — Результаты работы алгоритмов.

		dj38	eil51	qa194
Муравьиный	Найденное решение	7451	572	11657
	Затраченное время, сек	223	328	23147
Генетический	Найденное решение	7895	559	11367
	Затраченное время, сек	156	117	1164
Гибридный	Найденное решение	7115	498	10938
	Затраченное время, сек	263	448	29234
Имитации отжига	Найденное решение	7158	441	10506
	Затраченное время, сек	15	20	69
Пчелиной колонии	Найденное решение	6656	439	11695
	Затраченное время, сек	21	26	645
Оптимальное решение		6656	430	9352

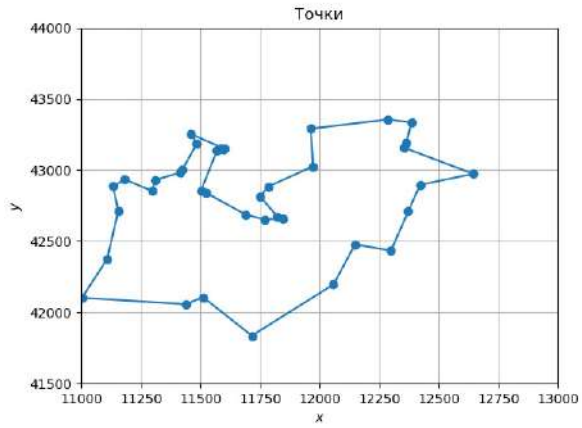


Рисунок 3.22 — dj38 после применения алгоритма имитационнго отжига

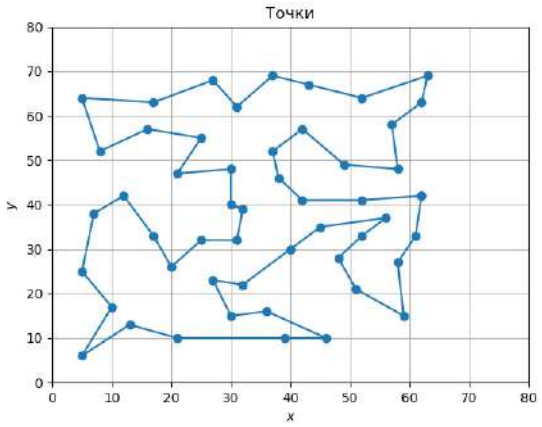


Рисунок 3.23 — eil51 после применения алгоритма имитационного отжига

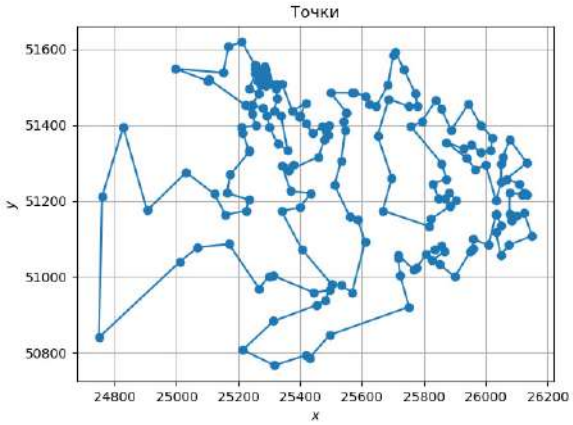


Рисунок 3.24 — Решение имитации отжига qa194

3.3.2 Построение максимального разреза, сравнительный анализ алгоритмов

Как показано выше, решение задачи $mTSP$ предваряет задачи кластеризации исходного графа, которая позволяет провести декомпозицию задачи в случае ее большой размерности. При наличии информации о распределении весов дуг и наличии информации о нежелательности использовать дуги большой длины эффективным является построение максимальных разрезов — *MAX-CUT*. Соответствующая модель является псевдобулевой.

Приведем сравнительный анализ алгоритмов нахождения максимального разреза. Задачу *MAX-CUT* можно сформулировать так: для графа $G = \langle V,E \rangle$ необходимо найти такие подмножества $R \subset V$ и $B = V/R$, чтобы число ребер (вес ребер) между R и B было максимально [75]. В работе исследуются алгоритмы приближенного решения задачи *MAX-CUT*. Так как задача построения максимального разреза является NP-полной, то для больших размерностей используются только приближенные методы решения [135].

Далее приведем сравнительный анализ эвристических алгоритмов и их комбинаций с переборными алгоритмами. Эвристика — это любая процедура, которая находит допустимое решение $\widetilde{x} \in X$. Желательно, чтобы $\widetilde{x}$ совпадало с оптимальным решением x^*, но для большинства эвристик можно только надеяться, что полученное решение будет близким к оптимальному [136].

Алгоритмы тестировались на графах размерностью от 10 до 200 с шагом 10, по 20 графов каждой размерности. Взвешенная матрица расстояний для них генерировалась случайным образом.

В вычислениях используется эвристический метод поиска в глубину Kaporis—Kirpusis—Stavropoulos, а из методов локального поиска — *Tabu-поиск*, жадный алгоритм и такие модификации жадного алгоритма как узловой жадный алгоритм и узловой жадный алгоритм с учетом предыдущего шага. Так же используются такие эвристики как метод глобального равновесного поиска (*GES*) и *GES+Tabu*. Эти алгоритмы начинают работу с некоторого начального решения. После чего на каждом шаге поиска решение заменяется на другое, более лучшее, найденное в окрестности текущего. Установлено, что отличительной

особенностью *Tabu-поиска* является то, что в памяти хранятся недавно найденные решения (список табу), который используется для того, чтобы избежать краткосрочного зацикливания в локальном минимуме. Алгоритм завершает свою работу после фиксированного числа итераций либо, если в течение нескольких шагов наилучшее решение осталось неизменным [136]. Так же используются эволюционные методы: GA, метод связывающих путей (PR) и некоторые эвристики GA: GA+*TSA*, GA+*Жадный*, GA+*Ядра*. Эволюционные алгоритмы применяются во многих достаточно сложных приложениях. Они являются итеративными, в них используются стохастические операторы к популяции, которые в начальный момент создаются случайным образом, при этом каждая особь является возможным решением. Используя функцию оценки, устанавливается пригодность особи в качестве решения [136]. В табл. 3.2 приведены результаты работы алгоритмов.

Таблица 3.2 — Среднее время работы и среднее отклонение алгоритмов

Алгоритм	Время работы (с.)
MAX-CUT Weight Global Equilibrium Search + Tabu	2465,4104
MAX-CUT Weight Path Relinking	19,397343
MAX-CUT Weight Global Equilibrium Search	1,3527289
MAX-CUT Weight Genetic Algorithm + Tabu	0,6371963
MAX-CUT Weight Genetic Algorithm	0,6335497
MAX-CUTWeightGreedy	0,6233704
MAX-CUT Weight Genetic Algorithm + Greedy	0,6135553
MAX-CUT Weight Genetic Algorithm + Cores	0,2267480
MAX-CUT WeightLorena	0,2267480
MAX-CUT Weight KaporisKirousisStavropoulos	0,0132480
MAX-CUT Weight Node GreedyMod1	0,0021864
MAX-CUT Weight Node Greedy	0,0021549
MAX-CUT WeightRandom	0,0000068

На рис. 3.25 построены графики среднего времени работы алгоритмов. По оси OX показана размерность графов, а по оси OY устанавливается время работы алгоритма в наносекундах.

Можно утверждать, что выбор алгоритмов существенно зависит от полноты информации по структуре графа и условиях на разрез [75].

Реоптимизация задачи многих коммивояжеров. В [63] показано, что исследование найденного оптимального решения при построении оптимального решения для задачи с новыми исходными данными (возмущенными начальными данными) связано с понятием реоптимизации и устойчивости. Реоптимизация в задачах $mTSP$ представляет особый интерес, т. к. добавление (удаление) вершин и/или дуг в большинстве практических случаев приводит к полиномиальным алгоритмам поиска оптимального маршрута. Многоагентность позволяет выделить кластеры с измененными данными и согласовывать решение на этих кластерах с решениями других агентов-коммивояжеров. Схема реоптимизации позволяет применять

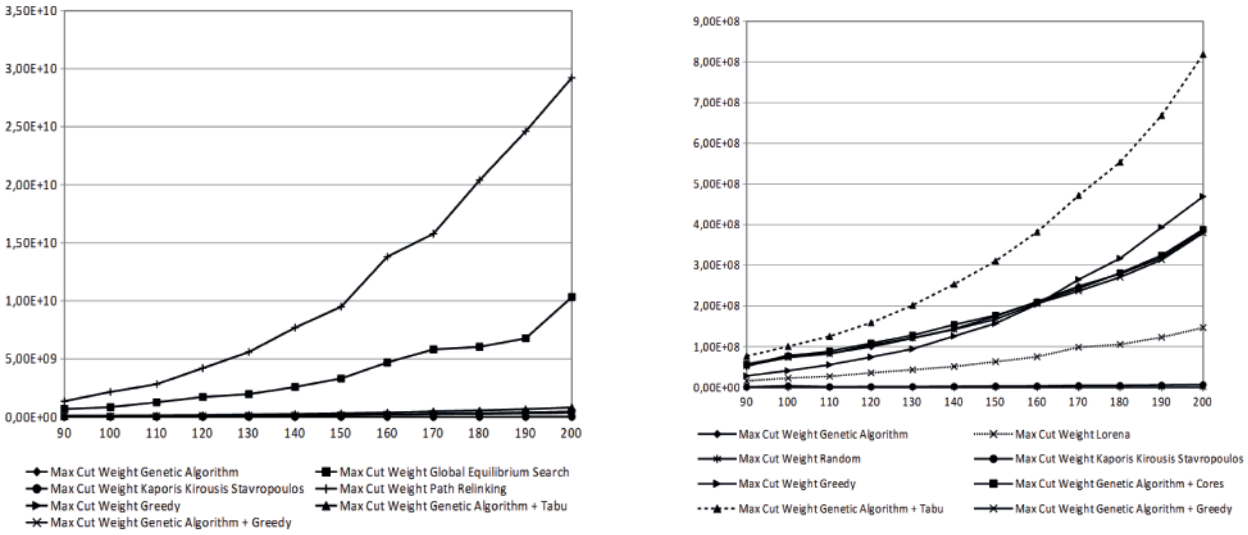


Рисунок 3.25 — Расчет среднего времени

иерархию *ИНС*, распределенных между агентами. При этом переобучению подлежит лишь часть нейронной сети.

Поиск оптимального устойчивого маршрута. Для проверки алгоритмов реоптимизации использовались стандартные библиотеки с координатами городов (532 вершины). С помощью программы *Concorde*, в которой использовался метод секущих для построения оптимальных маршрутов добавлялись вершины на границе области расположения вершин и внутри найденного контура. Как и предполагалось, для добавления вершин вне границы контура, устойчивость сохранялась (не менялась ранее найденная часть маршрута). Добавление вершины внутри контура приводило к резкому изменению нового контура. В случае проведения кластеризации с помощью максимального разреза, добавление или удаление вершины приводило к более устойчивым процессам, если резко не менялся сам разрез.

Выводы

1. Результаты по реализации алгоритмов данной главы опубликованы в работах [36; 91; 129].

2. Исследовано применение эволюционных многоагентных алгоритмов, их эффективность и возможность оптимизации для решения задач типа многих коммивояжеров. Кластеризация с помощью алгоритма максимального разреза используется для распределения вершин между агентами-коммивояжерами. Сравниваются алгоритмы: муравьиный, роя частиц, пчелиный, генетический, многоагентный жадный алгоритм и их комбинации. Показана перспективность метода кластеризации (в частности, основанного на построении максимальных разрезов) в сочетании с метаэвристиками. В предположении, что проведена предварительная обработка сети с учетом ее структуры, т. е. найдено разбиение графа сети на подграфы (кластеризация) задача построения маршрутов коммивояжера, которая реша-

ется на сети меньшей размерности. Сравнивается работа алгоритмов муравьиной колонии, имитации отжига, пчелиной колонии. Результаты по гибридным алгоритмам показывают возможность оптимизационной комбинации базовых алгоритмов. Тестирование алгоритмов на известных графах показало, что муравьиный алгоритм требует доработки, а алгоритм имитации отжига дает хорошее приближение к оптимальному решению. Результаты тестирования алгоритмов отражены в работах [91; 129].

3. Показано, что в прикладных задачах маршрутизации типа $mTSP$ предпочтительным является сочетание нескольких подходов, основанных на снижении размерности сети (кластеризация) и проекции исходной сети на более простую по своей структуре сеть, приводящую к полиномиальной сложности решения. На уровне программной реализации необходимо применять композиции алгоритмов (метаэвристик, генетических, роевых и др.) [36]. Рационально полиномиальными жадными алгоритмами находить предварительные маршруты, а затем их уточнять. В работе использованы для этого алгоритмы 2-Opt и эвристика имитации отжига.

Глава 4. Приложения знаниеориентированных моделей прикладной маршрутизации

4.1 Модели маршрутизации в чрезвычайных условиях

Реальные инфраструктурные сети являются сложными сетями большой размерности. В режиме чрезвычайных ситуаций структура сети может меняться с течением времени, что существенно усложняет применение классических методов решения задачи управления по выбору маршрутов, обеспечивающих доставку актуальных ресурсов. Рассмотрим проблематику задачи и алгоритмы маршрутизации на примере инфраструктуры Южного берега в условиях чрезвычайных ситуаций ($ЧС$).

Математическое моделирование процессов, связанных с предотвращением и ликвидацией последствий $ЧС$, например, землетрясений в местности, сочетающей горы и морское побережье, сопряжено с рядом специфичных особенностей самой структуры управления и принятия решений [18; 24; 91; 129; 136; 169]. Существующая организация системы управления ($СУ$) в условиях чрезвычайных ситуаций показывает их недостаточную эффективность [7]. В данном случае это не связано с какими-то недоработками $ЛПР$, а имеет объективные причины, и прежде всего, связанные с размерностью задачи управления сложной системой по ликвидации $ЧС$.

4.1.1 Проблемные ситуации и постановки новых задач

Математическое моделирование процессов, связанных с ликвидацией $ЧС$ порождает ряд новых постановок задач дискретной оптимизации, направленных на разрешение проблем по принятию решений в мобилизационных условиях.

Проблема 4.1. Размерность и сложность задач в условиях $ЧС$.

Необходимость быстрого принятия решения требует использования методов, алгоритмов и технологий декомпозиции. Можно говорить о рациональном, приемлемом (но не оптимальном) разделении систем на ряд локальных управляемых подсистем, обеспечивающих достижение общей цели.

С математической точки зрения наиболее подходящими в этом случае являются декомпозиционные потоковые методы, применяемые в сложных сетях. Ограничимся примером возможных ситуаций и выбором региона Большой Ялты для наглядности возникающих задач и алгоритмов маршрутизации по доставке ресурсов.

В задаче обеспечения водой в условиях землетрясения стратегическая ситуационная зона ($ССЗ$) территориально соответствует Большой Ялте с четко выделенными коллективами (бригадами, экипажами) на определенный период, за которыми закрепляется данная зона

как первоочередный и единственный объект обслуживания. Декомпозиция всей зоны между независимыми экипажами должна обеспечивать максимальную независимость по ресурсам (материальным, энергетическим, информационным). Здесь сочетаются задачи систематического районирования и кластеризации *ССЗ* на подзоны по объектам и бригадам.

Проблема 4.2. Распределение ограниченных ресурсов между выделенными территориями и ликвидационными бригадами *ССЗ*, обеспечивающих достижение общих целей.

Частичное решение этой проблемы связано с заранее установленной системой приоритетов и очередностью в зависимости от социальной значимости на основе углубленного анализа и с учетом прецедентного опыта.

Решается ряд задач анализа и синтеза *СУ ЧС*. Большинство задач синтеза структур различных классов сводятся к нелинейным задачам дискретной оптимизации и относятся в классу NP-трудных [24; 91; 169].

Согласно предварительно построенным сценариям развития *ЧС* (оптимистичным, пессимистичным, катастрофическим) осуществляются мероприятия по ликвидации *ЧС* в *ССЗ* на территории Большой Ялты. Ограничимся локальной задачей обеспечения водой. Рассмотрим, какие задачи необходимо решать на различных этапах по предотвращению и ликвидации *ЧС* на Южном Берегу Крыма.

Задача 4.1. Моделирование и анализ существующей сети источников воды, водопроводной сети, пожарных гидрантов, приоритетных потребителей воды (питьевой и технической), построение графовой инфраструктуры сети с привязкой к дорожной и другим сетям (электрическим, тепловым и др.). Привязка водной инфраструктуры к местности с помощью *ГИС*-технологий, Яндекс карт и т. п. Аналогичная задача ставится для других потоков.

Задача 4.2. Расчет потока в сети, расчета потока в подсети гидрантов, подсети резервных источников. Максимальный поток и поток при ограничениях на интенсивность источников, вплоть до нулевых значений для некоторых из них. Такие расчеты позволят принимать адекватные решения в случае *ЧС*. Экспериментальная проверка теоретических расчетов. Учет прецедентных случаев: разрыв сети, перекрытие части инфраструктуры (отключение части сети).

Задача 4.3. Задача расчета работоспособности сети при повреждении ее части. По выходу из строя (отсутствие воды в том числе) гидрантов, водозаборов технической воды для тушения пожаров, источников питьевой воды.

Задача 4.4. Маршрутизация бригад по ликвидации *ЧС* для существующей дорожной сети в случае потери некоторых маршрутов. Поиск независимых циклических маршрутов с одним или несколькими депо (задачи типа многих коммивояжеров).

Задача 4.5. Построение вариантов кратчайших путей от заданных вершин до приоритетных в стабильной ситуации и при ликвидации последствий *ЧС*.

Задача 4.6. Разработка расписаний проведения ликвидационных работ в зависимости от вариантов реализации разработанных заранее сценариев.

Задача 4.7. Облет сети. Составление наилучшего варианта выбора маршрутов коммивояжеров по результатам построения маршрутам облета сети (квадракоптер, вертолет).

В работах автора [30; 61] рассматривается специфика новых постановок задач построения кратчайшего пути транспортной задачи и некоторых других в условиях моделирования *ЧС*. Рассмотрим более подробно некоторые из приведенных задач (4.4, 4.5, 4.7).

4.1.2 Задачи маршрутизации по обеспечению водой в условиях чрезвычайной ситуации

Задача состоит в определении маятниковых и кольцевых маршрутов, например, для двух или m экипажей, обеспечивающих доставку воды к необходимым объектам. Каждый из них включает один или больше источников воды и определенную зону обслуживания объектов. Данная задача сводится к задаче m-коммивояжеров с множеством выделенных вершин-источников (депо, состоящее из нескольких вершин).

Алгоритм 4.1. *Алгоритм маршрутизации по доставке водных ресурсов экипажами*

1: *Разбить множество объектов-вершин на кластеры, приблизительно одинаковой размерности или согласно естественной для территории кластеризации (глава 3).*
2: *На каждом кластере решить задачу коммивояжера с обязательным посещением множества источников воды.*
3: *Получить множество маршрутов экипажей с рассчитанной стоимостью перевозок.*

Выделим случай одного источника и m коммивояжеров (экипажей). В этом случае в сети выделяется одна вершина-источник (депо), которую обязаны посетить все экипажи. Их маршруты распределены по выделенным зонам (кластерам) и могут быть как кольцевыми, так и маятниковыми, т. е. применима иерархическая кластеризация.

Реализация алгоритмов решения задачи многих экипажей-коммивояжеров. С практической точки зрения $mTSP$ может иметь множество особенностей. К примеру, максимально допустимая загрузка одного коммивояжера, существование нескольких баз, задержки при посещении объектов, ограниченное время работы и т. д. Будем использовать упрощенную постановку задачи. Ограничимся случаем сбалансированной $mTSP$. В этой постановке количество объектов (вершин) в маршруте каждого из коммивояжеров не должно сильно отличаться.

Для выбора алгоритмов решения $mTSP$ будем проводить вычислительный эксперимент на реальных данных для сравнения возможных методов решения $mTSP$ и выявления тенденций для продолжения исследований. Будем применять алгоритмы 3.1 и 4.2.

Алгоритм 4.2. *Сбалансированный маршрут $mTSP$*

1: *На заданном множестве точек (кластер) произвести построение первого приближенного маршрута. Здесь в основе лежит переход к полярным координатам и градация точек по возрастанию угла (алгоритм развертки).*

2: *Найти предполагаемый центр с помощью центра масс данного множества точек и вершину, которая является ближайшей к найденному центру масс — эту вершину и принять за базу коммивояжеров.*
3: *Распределить вершины, полученные на шаге один между m коммивояжерами. Построить такие маршруты, чтобы количество ребер в них отличалось не более чем на заданную величину (например, на* 1*).*
4: *Найти субоптимальный маршрут для каждого кластера, применяя алгоритм ближайшего соседа для нахождения начального приближения. По начальному приближению каждого маршрута произвести поиск субоптимального маршрута. На этом шаге используется вариация алгоритма* 2*-Opt и метода имитации отжига.*
5: *После получения субоптимальных маршрутов провести их визуализацию.*

4.1.3 Общие сведения о программном комплексе

Область применения: программный комплекс предназначен для выбора наилучших маршрутов $mTSP$ в случае структурных преобразований сложной инфраструктурной сети. Многоагентный подход в сочетании с упрощением структуры и кластеризацией сети позволяют получить предварительный набор маршрутов многих коммивояжеров, реоптимизация которых позволяет строить новые маршруты в случае изменения структуры сети. Программная реализация является решением задачи $mTSP$ для г. Ялты с прилегающими территориями в случае нескольких агентов-коммивояжеров. Многоагентность данных позволяет планировать реализацию функционала взаимодействия в MAC для имитации режимов *ЧС*.

Программный комплекс реализован на языке программирования *Python* версии 3.8. В расчетных модулях используются *opensource* библиотеки: *numpy*, *matplotlib*, *networx*, *numba*. Слой представления написан на *JavaScript* с использованием *nodejs*. Серверная часть реализована на основе *opensource* библиотеки *Express*. Клиентская часть для представления использует *React*, для отображения карт используется библиотека-обертка для сервисов Яндекс *react-yandex-maps*. Данная библиотека предоставляет удобный интерфейс для работы с открытым *API* Яндекса.

Поддерживаемые операционные системы: Windows, Linux, MacOs.

4.1.4 Логическая структура программного комплекса

Данный комплекс является многоуровневым приложением. В архитектуре можно выделить три основных слоя: расчетный, главный и представление (рис. 4.1). Такая структура обеспечивает гибкое и удобное масштабирование в процессе использования продукта.

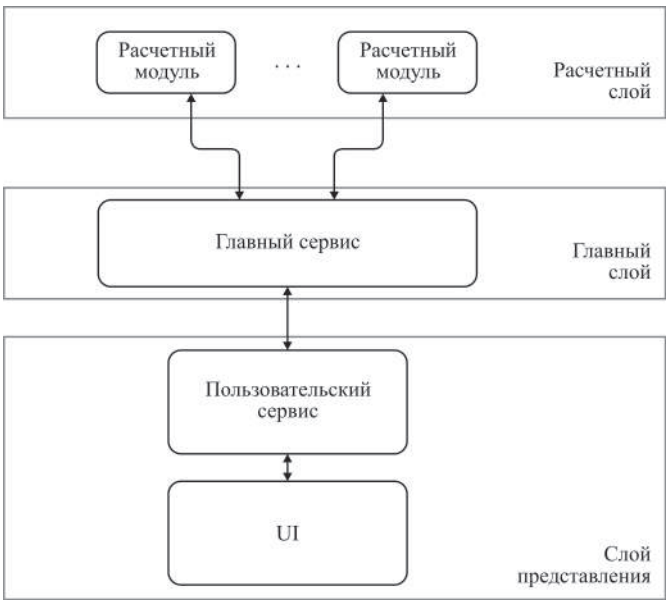


Рисунок 4.1 — Архитектура программного комплекса многоагентной инфраструктурной маршрутизации

Главный слой представляет собой основной модуль, который отвечает за распределение задач, мониторинг их выполнения, оценку результатов. Этот слой получает на вход множество координат точек, которое отправляется в один или несколько расчетных модулей. Допускаются конвейерная обработка множества, то есть сначала данные обрабатываются первым расчетным модулем, потом вторым и так далее. На выход слой отдает последовательность, которая соответствует найденному субоптимальному маршруту для заданного числа коммивояжеров.

Расчетный слой состоит из множества расчетных модулей (общая схема приведена на рис. 4.1), каждый из которых реализует определенный алгоритм. Модуль на вход получает последовательность координат точек. В теле происходят необходимые вычисления и преобразования. На выход отдается преобразованная последовательность координат.

Слой представления является клиент-серверным приложением. Клиентское приложение общается с главным слоем только через посредника, это позволяет всегда контролировать данные, которые получит клиент. На данном этапе сервер на выход отдает последовательность координат для каждого из объектов в городе Ялта и множество координат, которые соответствуют связи по автомобильной дороге, для каждой пары объектов.

Такая архитектура обеспечивает легкость добавления новых расчетных блоков. Расчетному модулю не нужно ничего знать о том, как данные будут выглядеть для пользователя или какие модули будут дальше обрабатывать полученное множество. Это выполняется в главном слое. Аналогично, главный слой не содержит алгоритмы маршрутизации и внутренней реализации каждого из расчетных модулей или слоя представления. Этот слой оперирует данными и следит за тем, чтобы они попали туда, куда нужно.

4.1.5 Структура данных

На вход подаются следующие данные:

pathsList — список путей к файлам с данными. Представляет собой список путей в файловой системе к файлам с данными, файлы данных должны соответствовать структуре, описанной выше, таким образом можно запускать расчет для нескольких независимых файлов данных в рамках одного запуска;

resDirectory — директория, в которую будут сохранены результаты тестов (строка с путем в файловой системе к директории в которую будут помещены результаты расчетов);

specMark — строка, уникальная метка, которая добавляется к каждому названию (строка, служит уникальным идентификатором для тестовых групп, это необходимо чтобы разграничивать результаты нескольких запусков в одной директории);

k — количество коммивояжеров для которых ищется решение;

testAmount — количество запусков для каждого файла с данными. Это необходимо для поиска наилучшего решения из нескольких запусков алгоритма на одном и том же множестве данных;

mode — строка из ['polar', 'kmeans'], соответствует алгоритму кластеризации, который будет применен при подготовке данных для поиска решений.

На вход подается текстовый файл с множеством вершин, описанных следующим образом:

1 34.14384102614102 44.48270535818645
2 34.17890309323493 44.50948110556448
3 34.24070444907345 44.51218681595662
4 34.23959133233223 44.51200640702147
5 34.23887250031624 44.51190276756466
....

Здесь первое число — это индекс вершины, счет начинается с 1. Второе и третье число — координаты вершины *longitude* и *latitude* соответственно.

На выходе приложение отдает *JSON*-файл со схемой:

```
Array<
{
    ‘‘tour’’: Array<number>,
    ‘‘distance’’: number
}
>
```

Пример выходных данных:

```
[
{
```

```
    ‘‘tour’’: [15, 383, 16, ..., 15],
    ‘‘distance’’: 2376.3971372556703
},
{
    ‘‘tour’’: [15, 123, 23, ..., 15],
    ‘‘distance’’: 2376.3971372556703
}
]
```


4.1.6 Структура расчетных модулей

На данный момент в программном комплексе «Программа многоагентной инфраструктурной маршрутизации» реализованы пять расчетных модулей.

1. Построение начального приближения. Задано множество точек P, нужно найти первый приближенный маршрут, который ляжет в основу разбиения на подмножества для каждого коммивояжера. В одном из алгоритмов в основе лежит переход к полярным координатам и градация точек по возрастанию угла относительно центра.

2. Поиск предполагаемого центра. Для этого находится центр масс данного множества точек и вершина, которая является ближайшей к найденному центру масс. Эта вершина в дальнейшем считается базой коммивояжеров.

3. Распределяем вершины, полученные на шаге один, между k коммивояжерами. Важно учесть, что необходимо построить такие маршруты, чтобы количество ребер в них отличалось не более чем на 1.

4. После получения k маршрутов строится оптимальный маршрут для каждого множества точек, т. е. решается задача одного коммивояжера для заданных вершин. Здесь применяется алгоритм ближайшего соседа для нахождения начального приближения.

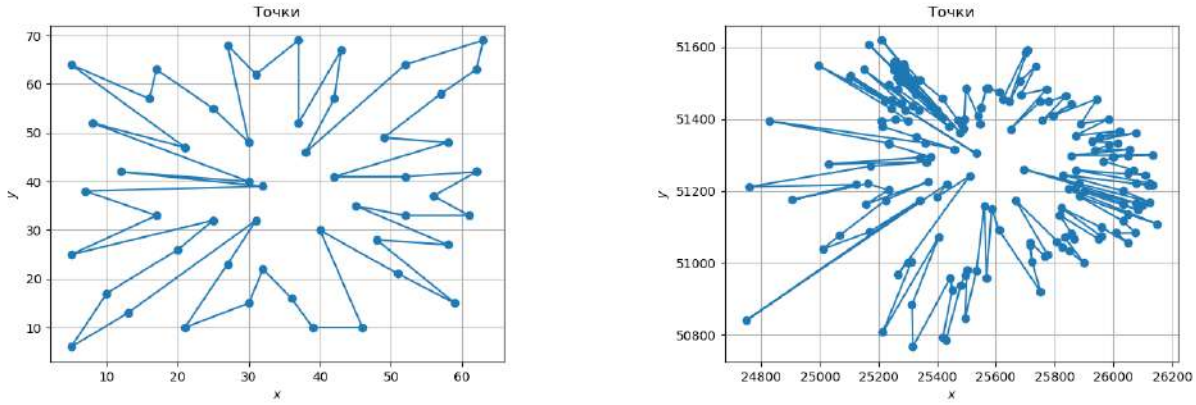
5. Поиск субоптимального маршрута (считается, что начальное приближение каждого маршрута сформировано). На этом шаге используется вариация алгоритм 2-Opt и метод имитации отжига.

Рассмотрим подробнее структуру каждого из указанных модулей.

1. **Алгоритм построения начального приближения**, является хорошим приближением для задач, определенных в декартовом пространстве. В основе лежит использование полярных координат. После приведения к полярным координатам множества вершин, появляется возможность упорядочить их в порядке возрастания угла φ. Дано множество декартовых координат $x_i^{'}$, $y_i^{'}$ при $i = 1...N$, узлов (пунктов, точек). Путем линейных преобразований $x_i' = x_0' + \lambda x_i$, $y_i' = y_0' + \lambda y_i$ переводим их в единичный квадрат. Введем полярную систему координат r, φ с началом в его центре. Точки нумеруем с ростом угла φ. Полученную нумерацию точек закрепляем и этот вариант считаем начальным приближением. Блок-схема алгоритма приведена на рис. 4.2. Пример работы алгоритма показан на рис. 4.3.

Рисунок 4.2 — Блок-схема алгоритма построения начального приближения

2. **Алгоритм нахождения оптимальной базы для многих коммивояжеров**. База является общей точкой для всех коммивояжеров, с нее они начинают свой маршрут (например, место нахождения экипажей $МЧС$ и множество точек, которые нужно объехать). Для поиска вершины соответствующей базы находится центр масс данного множества, а затем вычисляется ближайшая вершина к найденному центру. На предыдущем этапе координа-



а) б)

Рисунок 4.3 — Пример работы алгоритма построения начального приближения: а) для 51 вершины; б) для 194 вершин

ты центра масс были найдены (X_{mid} и Y_{mid}). Остается только найти ближайшую вершину. Здесь используется стандартное декартово расстояние, так как мы работаем в двумерном декартовом пространстве.

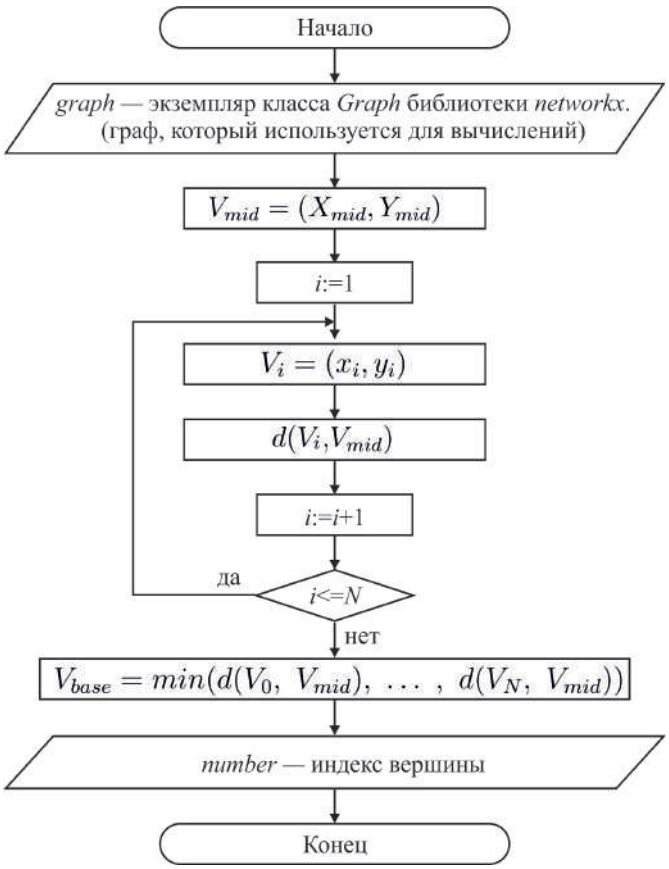


Рисунок 4.4 — Блок-схема алгоритма нахождения оптимальной базы для многих коммивояжеров

3. **Алгоритм распределения вершин между k коммивояжерами**.

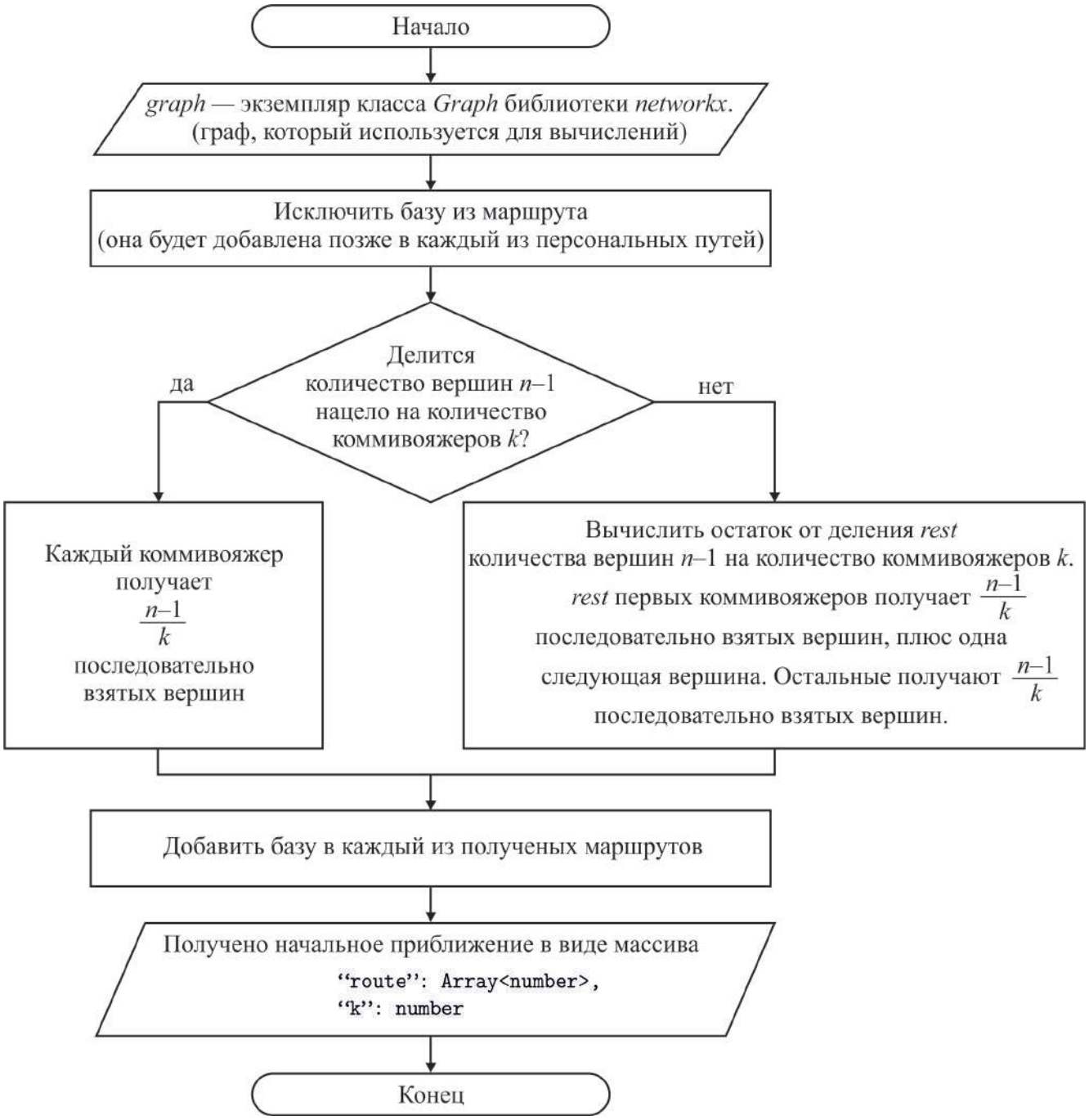


Рисунок 4.5 — Блок-схема алгоритма распределения вершин между k коммивояжерами

4. ***Алгоритм ближайшего соседа***.

Вход: *graph — экземпляр класса Graph библиотеки networkx (граф, который используется для вычислений).*

Выход: *упорядоченная последовательность индексов вершин.*

Схема алгоритма:

1: *Взять случайную вершину графа. Поместить индекс вершины в список. Отметить ее.*

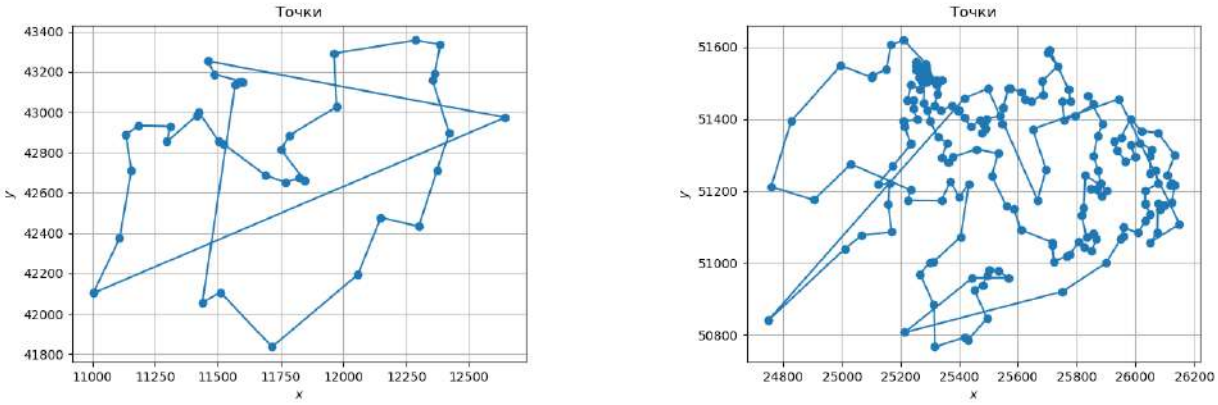
2: *Проверить есть ли не отмеченные вершины.*

3: *Вычислить расстояние до каждой из неотмеченных вершин графа.*

4: *Найти вершину расстояние до которой минимально. Поместить индекс вершины в список. Отметить ее.*

5: *Вернуться к шагу* 2.

Очевидно, что построенные так маршруты не являются оптимальными (рис. 4.6).



а) б)

Рисунок 4.6 — Пример работы алгоритма ближайшего соседа: а) граф из 38 вершин; б) граф из 194 вершин

5. **Алгоритм нахождения оптимального маршрута.** За основу взят известный алгоритм 2-*Opt*. Он принадлежит к множеству алгоритмов локального поиска. Впервые алгоритм опубликовал Г. А. Круз в 1958 г., подробный ход решения описал М. М. Флуд. Главный принцип алгоритма состоит в том, чтобы взять произвольный маршрут, который, скорее всего, имеет самопересечения и изменить порядок обхода на такой, который не будет пересекать сам себя. Для того, чтобы получить оптимальное решение, необходимо перебрать всевозможные маршруты и их замены. В реальных сетях при размерности задачи около 500 вершин полный перебор начинает занимать ощутимое время, поэтому на практике применяется ограничение по итерациям. Этот подход позволяет отыскать хорошие субоптимальные решения. Как правило, производится несколько запусков и уже из них, лицом, принимающим решение, выбирается лучший результат.

Схема алгоритма:

Вход: *points — последовательность индексов вершин.*

Выход: *Array<number> — упорядоченная последовательность индексов вершин.*

1: *Выбрать участок маршрута, на котором будем производить оптимизацию. Обозначим его* r.

2: *Выбрать случайные точки* i *и* j *из диапазона* $[0 \dots N]$, *где* $N = |r|$ (*мощность множества* r).

3: *Проверить условие* $i < j$. *Если оно не выполняется, то поменять значения местами.*

4: *Взять часть* r, *начиная с* 0*-го элемента и заканчивая* i*-м. Обозначим ее* r_1.

5: *Взять часть* r, *начиная с* i*-го элемента и заканчивая* j*-м. Обозначим ее* r_2.

6: *Взять часть* r, *начиная с* j*-го элемента и заканчивая* N*-м. Обозначим ее* r_3.

7: *Вернуть новый маршрут*: $r_1 + r_2 + r_3$.

Если работа происходит над замкнутым маршрутом или важно начать или закончить маршрут в конкретной вершине, то следует исключить данную вершину из маршрута перед передачей ее в алгоритм. Это связано с тем, что после изменения порядка следования по вершинам, будет получен ошибочный маршрут.

Этот алгоритм лежит в основе алгоритма имитации отжига, который хорошо себя показывал в различных исследованиях. В работе была использована модификация этого алгоритма для размерности меньше либо равной 100 вершинам. В этом случае, помимо 2-*Opt* произведен поиск лучшей замены, то есть, если классический 2-*Opt* применяет любую замену, которая улучшает длину, то в модификации будут отлавливаться только самые лучшие.

4.1.7 Результаты экспериментов на инфраструктурной сети Большой Ялты

Самостоятельной является задача программной реализации *ИА* и *МАС* по поиску маршрутов в меняющихся условиях. В качестве примера рассмотрим инфраструктурную сеть дорог и объектов водообеспечения. В качестве исходных используются данные на Яндекс.Картах (*ГИС*). Объекты, расположенные на ней, представлены в виде информации на разных слоях карты. Многослойность картографических данных позволяет получать более целостные и полезные, с практической точки зрения, результаты. В данном примере на карте представлена инфраструктура и координаты объектов водоснабжения Большой Ялты (гидранты, колодцы и т. п.). В положении *ЧС* остро стоит вопрос быстрого доступа к водосодержащим объектам. Моделирование *ЧС* предполагает изменение маршрутов в случае выхода из строя ряда водных объектов, завалов на дорогах или расконсервирование резервных источников воды, ее доставка для компенсации утраченных.

Для построения маршрутов, наземные объекты можно посетить, передвигаясь по земле (например, автомобилем или пешим ходом). Второй способ — перемещение по воздуху (например, дрон или вертолет). Таким образом, явно можно выделить два слоя в данной задаче, и каждый слой имеет свои принципиальные особенности, обусловленные физическими возможностями в каждом из пространств. В воздушном слое можно в расчет принимать *кратчайшее расстояние по прямой* (дистанция между точками). В наземном слое требуется брать в расчет существующие дороги и подъезды к объекту. Воздушный слой, конечно, предоставляет больше возможностей и формирует полный неориентированный граф между точками. Решение $mTSP$ на такой сети затем проецируется на реальную сеть S (обобщение алгоритма 3.1).

Также для упрощения визуализации отображения на схематическом рисунке построенных маршрутов можно взять проекцию реальных точек с дорогами на преобразованную сеть S_i, $i \in I$. Здесь каждое ребро будет иметь вес, соответствующий кратчайшему расстоянию между объектами. Данный подход позволяет избежать нагромождения на графике и предоставляет возможность просмотреть предварительные приближенные результаты.

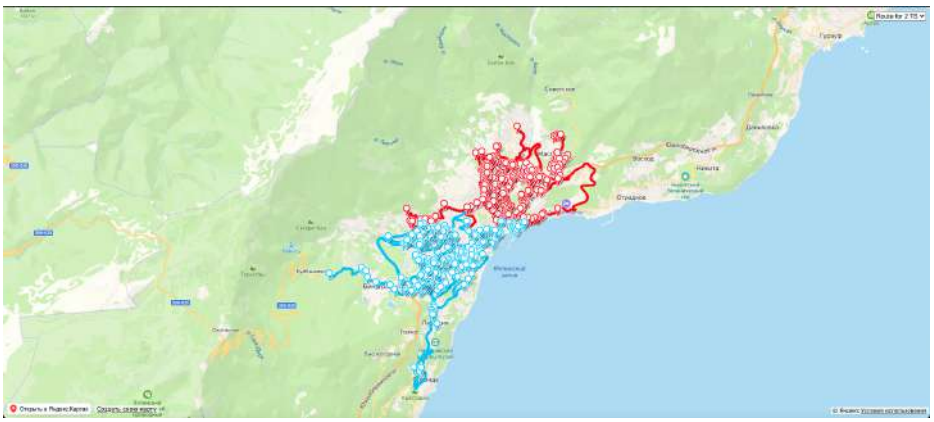


Рисунок 4.7 — Решение для двух коммивояжеров в городе Ялта с прилегающими территориями

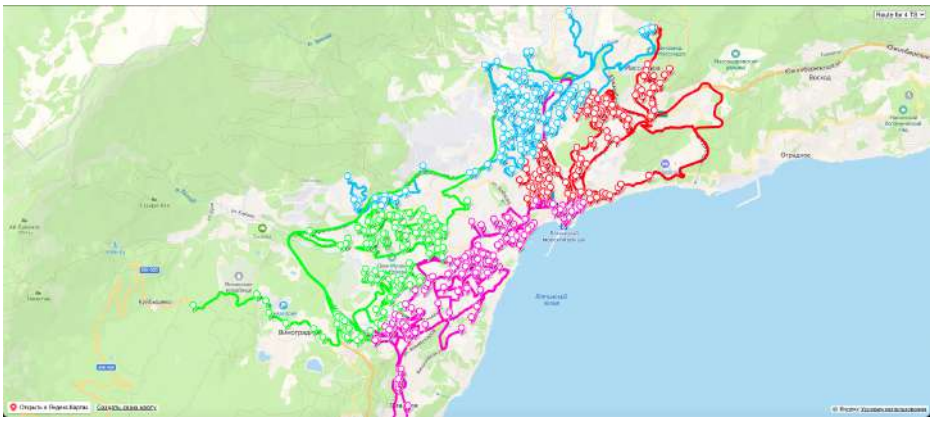


Рисунок 4.8 — Решение для четырех коммивояжеров в городе Ялта

В критической ситуации требуется максимально быстрый расчет по предварительным, начальным, приближенным решениям задачи $mTSP$, на основании которого производятся уточнение, исключение и добавление объектов. Реализован подход, согласно которому исходной сети ставится сеть с графом облета сети (проекция на плоскость), в котором каждое ребро имеет вес, соответствующий кратчайшему расстоянию между объектами (алгоритм 3.1).

На рис. 4.7 представлено решение задачи $mTSP$ для Ялты с прилегающими территориями в случае двух коммивояжеров. На рис. 4.8 представлено решение $mTSP$ (4 коммивояжера) для центрального кластера (Ялта).

Грубое решение на кластерах коммивояжеров улучшается с помощью вариации алгоритма 2-Opt и алгоритма имитации отжига. Многоагентность данных позволяет планировать реализацию функционала взаимодействия в MAC с уже полученными данными для имитации режимов $ЧС$. Предполагаемое приложение должно содержать реализацию интерфейса пользователя, который содержит инструменты изменения элементов графа сети, выбор и модификацию алгоритмов, позволяющих производить лишь локальный перерасчет данных (реоптимизация).

4.2 Многоагентные модели в геоинформационных системах

4.2.1 Программный комплекс выбора наилучших туристических маршрутов по Крыму

Задачи маршрутизации в сложных сетях, как уже было указано, являются объектом исследований, как в теоретических работах, так и в приложениях. В данном случае рассматриваются сложные сети, соответствующие рекреационной деятельности на выделенной территории. Они включают инфраструктуру дорог, услуг и разнообразных объектов, интересующих рекреантов (туристов). Такие объекты можно называть достопримечательностями. Они имеют разнообразное наполнение: объекты природы, исторические объекты, здания, музеи и т. п. С каждой достопримечательностью связана некоторая организация: собственник; предприятия, предоставляющие экскурсионные услуги, услуги размещения и т. п. Поэтому для данного случая формализация задач на таких сетях оперирует терминологией сетей, графов: *достопримечательность — организация — объект — вершина — узел*. Текущая сеть является частью некоторой большой сложной сети, соответствующей данной территории. Ее фиксация зависит от полученной информации о желаемых для посещения достопримечательностей и инфраструктуры, связывающей эти объекты. Возможно, запрос на дополнительную информацию позволяет выявить лучший вариант сети. Тем самым задача оптимальной маршрутизации в этом случае будет отличаться от классических постановок задач дискретной оптимизации на графах: задачи коммивояжера $mTSP$, нахождения кратчайшего пути, задачи инкасатора, почтальона и т. п. Используются итерационный процесс построения рациональных маршрутов, системы уточняющих запросов, сужающих область неопределенность [72] и оптимизация полученных маршрутов. Прецедентами является база ранее построенных маршрутов, множество которых частично пересекающихся с маршрутами, построенными по запросам пользователей.

Планирование маршрутов, как правило, предполагает составление маршрута посещения объектов заранее, до начала осуществления экскурсии. Но часто такое планирование необходимо осуществлять в реальном времени. И чем быстрее получен результат, чем больше вариантов можно предъявить для выбора пользователю, тем лучше. Это накладывает некоторые дополнительные требования на реализацию планирования маршрутов в конкретном веб-сервисе (программном приложении). Пользователем задаются предпочтения по выбору достопримечательностей, которому предваряет поиск достопримечательностей (организаций, объектов, вершин).

Программный комплекс выбора наилучших туристических маршрутов ($ПКВНТМ$) по Крыму реализуется как планировщик маршрутов по достопримечательностям на основе сервиса карт. Для поиска достопримечательностей можно использовать поисковики (Google, Яндекс), прочитать отзывы на специализированных сайтах (TripAdvisor), забронировать отели через сайты-агрегаторы отелей (Booking), для оптимального передвижения по городу

использовать сервисы карт (Google Maps, Яндекс.Карты, 2*ГИС*). На каждую область туристической деятельности приходится какой-нибудь сервис. Кроме одного — планирования. Даже с учетом всех технологий, с учетом систем рекомендаций, людям, отправляющимся в путешествие, многое приходится держать в голове или записывать. Процесс планирования познавательного отдыха (или работы) достаточно длительный и утомительный, а его результат, далеко не оптимальный.

Программный комплекс предназначен для реализации следующей цели: на основе данных о достопримечательностях, полученных из открытых достоверных источников, а также на основе данных пользователя с помощью разработанного сервиса, составить оптимальный маршрут на несколько дней по достопримечательностям, с учетом их интервалов посещаемости и желаемого времени посещения. Структура *ПКВНТМ* приведена на рис. 4.9

Рисунок 4.9 — Структура *ПКВНТМ*

В силу большого разнообразия привлекательных объектов в качестве региона для демонстрации выбрана Республика Крым, которая является показательной территорией для тестового анализа соответствующего веб-приложения. Аналогов подобного приложения пока нет. Есть сайты для построения оптимального маршрута по нескольким точкам, однако они рассчитаны на доставку (`http://blog.ulnar.ru/`,`https://poncy.su/`). Есть сайты, которые строят пешеходные маршруты по достопримечательностям (`https://sightsafari.city/`), однако они рассчитаны именно на краткий обход по достопримечательностям внутри одного города, но никак не посещению мест региона. Ближайшими аналогами разрабатываемого приложения как раз являются картографические сервисы Google Maps и Яндекс.Карты. Они позволяют строить маршрут из нескольких точек только лишь в указанном порядке, однако имеют ряд нужных нам преимуществ, которых лишены конкуренты — отмеченные

на карте достопримечательности с достоверной информацией от владельцев, высоко интерактивный интерфейс.

Рассмотрим специфику базовых блоков. Часть возможностей картографических сервисов поставляется в виде бесплатного *API* (используется в веб-приложении).

4.2.2 Сбор и хранение данных об организациях

Поисковые организации, такие как Google и Яндекс, имеют сервисы, которые позволяют владельцам бизнесов регистрировать свои организации на карте, вводить туда актуальную информацию о времени работы, стоимости билетов, вебсайты организаций и т. д. У Яндекса, например, свой бизнес можно зарегистрировать через Яндекс.Справочник. Для бизнесов регистрация на таких площадках давно стала стандартом для продвижения, поэтому у организаций часто можно увидеть заполненную контактную информацию, время работы, фотографии, отзывы и т. д. Часть этих данных доступна через бесплатный *API*.

В *ПКВНТМ* использованы сервисы Яндекса (лучше индексируют территорию стран *СНГ*), а именно «*API* Поиска по организациям» от Яндекса. Чтобы обратиться к *API* справочника, нужно сделать *HTTP*-запрос к серверам Яндекса, ответом является список организаций до 500 штук в *JSON*-формате. Такой способ получения информации об организациях прост, содержит актуальную информацию, возвращает информацию, достаточную для идентификации места, отображения контактных данных и определения местоположения. Однако, он имеет ряд недостатков: ограничение на бесплатное пользование; нет информации для отображения фотографий, отзывов; нет информации, для сортировки организаций по значимости.

Для удобства организации разбиты на категории. Обращением за данными организаций является *HTTP*-запрос: `https://search-maps.yandex.ru/v1/`
? apikey=<ключ>
& text=<поисковый запрос>
& lang=<язык ответа>
& [type=<типы объектов>]
& [results=<количество результатов в ответе>],
где
apikey — это ключ, по которому сервис считает ограничения на количество запросов в сутки;
text — запрос вида «название категории, Республика Крым»;
lang=ru_RU;
type=biz (организации);
results=500 — ограничение ответа в 500 организаций.

Выполняя подобные запросы к каждому типу организаций, получаем массив организаций. Для каждой категории достопримечательностей ответы сохраняются в *JSON*-файлах.

4.2.3 Структура интерфейса

Для разработки приложения выбраны технологии React (приложение имеет высокую интерактивность, размещается на одном экране, имеет множество состояний) и TypeScript (нужен для контроля данных, т. к. форматы данных организаций довольно массивны).

Интерфейс приложения приведен на рис. 4.10

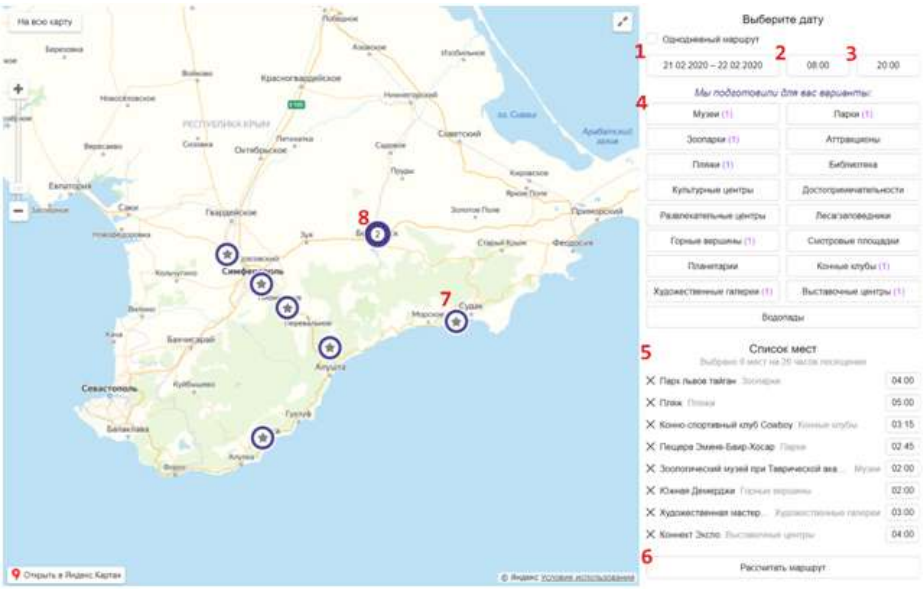


Рисунок 4.10 — Интерфейс программы выбора туристических маршрутов по Крыму

Экран приложения разделен на две области: слева карта (реализована в Яндекс.Карты, карты берут на себя реализацию перемещения, масштабирования, вызова событий, оптимизированного отображения точек, кластеризации точек и отрисовки маршрута); справа текстовый интерфейс приложения, который позволяет осуществить:

— выбор даты для построения маршрута;

— выбор времени начала маршрута;

— выбор времени конца маршрута — максимальное время, когда пользователь готов оказаться в конечной точке;

— список категорий достопримечательностей;

— список организаций, отобранных пользователем (каждый элемент списка содержит название, категорию, время, которое пользователь хочет там провести);

— расчет оптимального маршрута, при условии, что выбрана дата, время и организации (если не выбрана ни одна категория, на карте отображаются лишь точки, отмеченные самим пользователем).

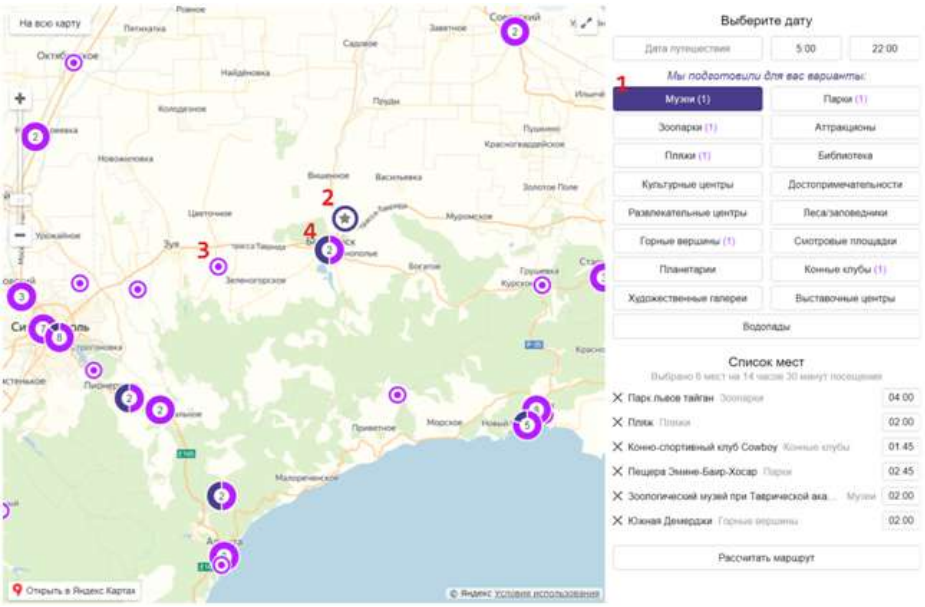


Рисунок 4.11 — Выбор пользователем категории достопримечательностей

4.2.4 Расчет оптимального маршрута по выбранным достопримечательностям

Рассмотрим, принцип взаимодействия с алгоритмом построения маршрута.

На вход алгоритма подаются данные:

массив времени начала и конца маршрута для каждого дня;
время начала в секундах;
время конца в секундах;
начальная и конечная точка маршрута;
массив достопримечательностей, выбранных пользователем;
координаты организации;
идентификатор;
время в секундах, сколько пользователь хочет там провести;
массив времени работы для каждого дня (первый массив);
в одном дне может быть несколько разбиений из-за перерывов;
время начала работы в секундах;
время конца работы в секундах.

На выходе алгоритм возвращает данные следующего формата:

начальная и конечная точка маршрута;
массив маршрутов по дням;
время начала маршрута в секундах;
массив посещения достопримечательностей по порядку;
идентификатор организации;
время начала посещения;
время конца посещения;
время ожидания открытия организации (можно приехать заранее);

координаты организации;
расстояние в метрах от предыдущей точки;
время окончания маршрута в секундах;
расстояния от последней организации;
массив организаций, которые не вместились в маршрут;
идентификатор организации;
координаты организации.

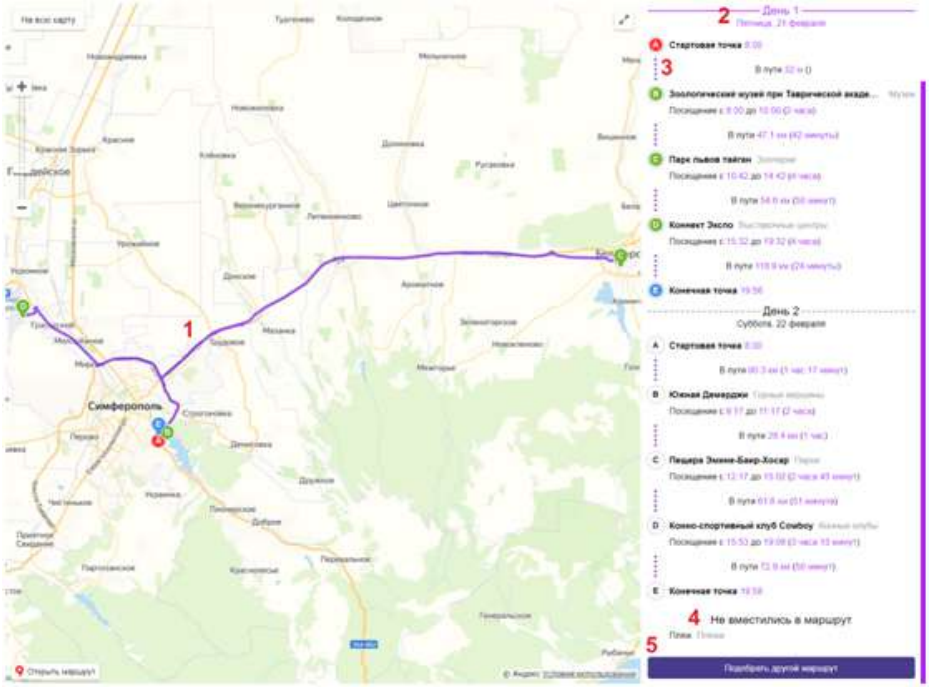


Рисунок 4.12 — Отображение маршрута на карте

Рассмотрим особенности алгоритма:

— учитывая длительность переездов и время посещения достопримечательностей (вершин), в один день может поместиться не более 5 достопримечательностей;

— можно пренебречь посещением (одной) достопримечательности для оптимальности маршрута в целом, т. е. алгоритм умеет отбрасывать максимально невыгодные вершины, дает возможность вернуть несколько путей с разными вершинами.

В данной программе вместо алгоритма *TSP* с окнами реализации выбран алгоритм обхода графа в ширину. Даже для большого количества вершин (порядка 50 достопримечательностей на все дни), он подходит под все данные параметры.

Алгоритм обхода в ширину рассматривает все возможные варианты, отбрасывая при этом ветви с невозможными вариантами как можно раньше, чтобы ускорить подсчет. Также к его плюсам можно отнести, что он не разделяет маршрут на отдельные дни, а производит расчет для всего маршрута, как для единого целого. Это позволяет ему не жертвовать точностью вычислений и получить наилучший результат.

Несмотря на то, что алгоритм в аппроксимации равен полному перебору путей, из-за оптимизаций, наличия расписаний и длинных переездов, он достаточно быстро (за время

обновления информации на Яндекс.Карте) находит результат для маршрута из десятка вершин.

4.3 Технологии многоагентной маршрутизации и меметика социальных сетей

4.3.1 Интеллектуализация обработки информации социальных сетей

Содержание данного раздела отражает результаты автора по сетевым (графовым) структурам и специфике маршрутизации, опубликованные в работах [185; 186; 211] и относятся к тематике распространения мемов в социальных сетях. Алгоритмы и технологии решения задач на графах находят применение при исследовании распространения мемов в графах социальных сетей. Формируется междисциплинарное направление исследований — меметика. Мемы являются одним из возможных источников вирусной информации. Задача определения влияния деструктивной информации, распространяемой в социальных сетях на ее потребителей и распространителей (активных агентов сети) связана со сложностью моделирования критериев оценки влияния, а также с определением кластеров. Здесь мемы рассматриваются как инструменты изучения социальных сетей. С мемами связываются производство, распространение, влияние, моделирование, анализ, оценка, противодействие, управление процессами преобразования деструктивной информации.

Социальные сети обладают огромным ресурсом неиспользуемой информации. Визуальная информация (изображения) сопровождается текстом и разными активностями (лайки, комментарии и пр.), отображающими динамику восприятия агентами-потребителями этой информации. Кластеризация потребителей, оценка системы влияния, управления влиянием для достижения коммерческих, политических и других целей являются актуальными задачами интеллектуализированного подхода обработки информации социальных сетей.

Исследование процессов распространения информации в сложных сетях конкретизируется на распространении мемов в социальных сетях. Существенную роль играют структурные факторы сети, которые способствуют или препятствуют распространению мемов. Обходы кластеров сообществ социальных сетей (маршрутизация) определяют пути распространения мемов по сообществам. Субъекты сети, как правило, входят в тесно связанные группы, которые способствуют вирусному распространению близких к ним мемов. Таким образом, важными для управления и прогнозирования распространения мемов являются структурные параметры кластеризации, локальные меры близости к влиятельным источникам распространения. Выявление агентов, принадлежащих к активным сообществам, может служить признаком распространения мемов. Агент (вершина сети), степень которого наибольшая, может быть активным распространителем вирусной информации, но агент, являющийся членом пересекающихся кластеров (сообществ) имеет больше возможностей для распространения мемов. Одной из целей в сетях является предсказание эффективности про-

цесса распространения, т. е. насколько быстро и насколько широко заражаются вершины сети (узлы-агенты). Обычно исследуется, какие узлы наиболее влияют на процесс распространения информации, что позволяет их контролировать (иммунизация, атака на эти узлы). Для выявления перекрывающихся кластеров (чередующихся, вложенных) [252] используется алгоритм линк-сообществ [145].

Рассмотренные в работе алгоритмы многоагентной маршрутизации на сложных сетях используются в *ИС* «Memometrix» — программный комплекс для анализа потока интернет-мемов. В этот комплекс входят компоненты для автоматического сбора, структурирования и анализа интернет-мемов.

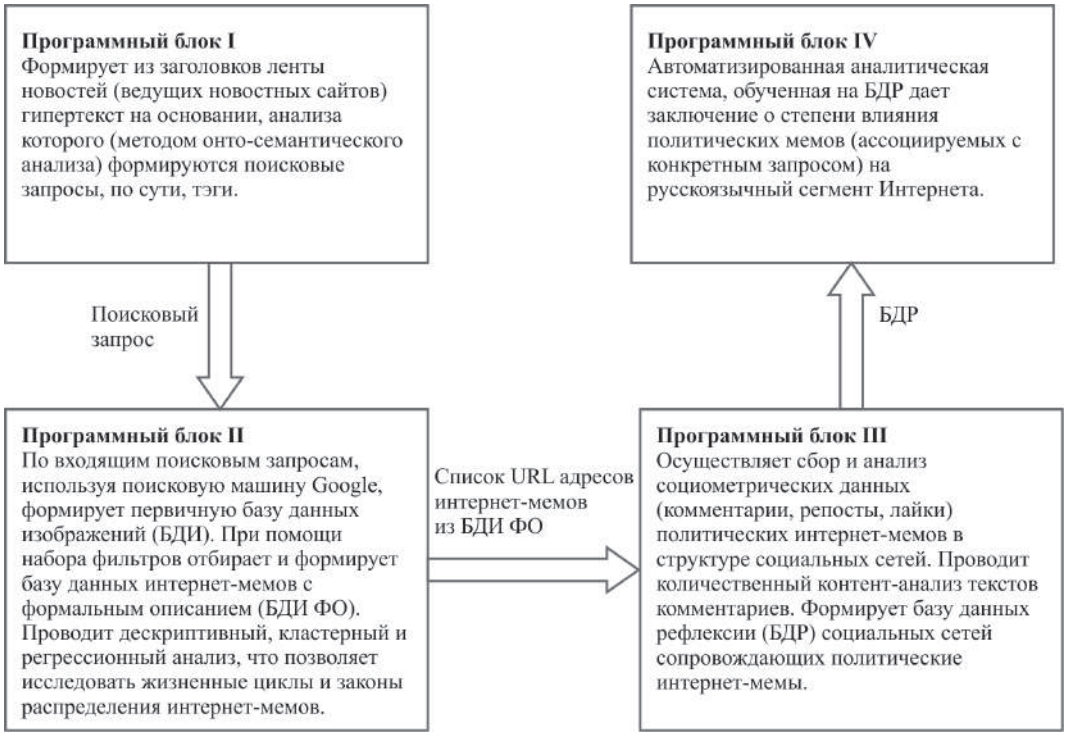


Рисунок 4.13 — Общая структура программного комплекса «Memometrix»

При разработке использованы языки программирования *C#*, *JavaScript* и *Python*.

На вход *ИС* «Memometrix» подаются запросы на мемы, отвечающие актуальным событиям, которые отражаются в заголовках новостных лент.

Схема формирования поисковых запросов:

1. Осуществлять сбор заголовков новостных сообщений с указанных интернет-ресурсов (например, `https://russian.rt.com/news`, `https://echo.msk.ru/news`, `https://yandex.ru/news/rubric/politics` и пр.). Сбор осуществляется в течение дня.

2. Собираемые в течение дня заголовки новостных сообщений на протяжении 5-ти дней [понедельник — пятница] объединяются в единый гипертекст.

3. Сформированный гипертекст подвергается онто-семантическому анализу. Для реализации онто-семантического анализа используется технология *Word2Vec*, позволяющая представить каждое слово из гипертекста в виде n-мерного вектора. Далее для оценки степени близости слов используется мера *Cosine* (косинусная близость) либо другие меры (*Euclid*, *Correlation*, *Jaccard*).

Следующая подзадача — визуализация полученной модели и анализ результатов. Полученная в результате преобразований модель содержит вектор чисел для каждого из слов входящего гипертекста. Простейшим способом визуализации является представление этих слов в виде точек на плоскости. Однако размерность полученных векторов превышает 100 (может достигать 1000). Для построения точек из n-мерного пространства на плоскости используется метод главных компонент (PCA), выделяются две.

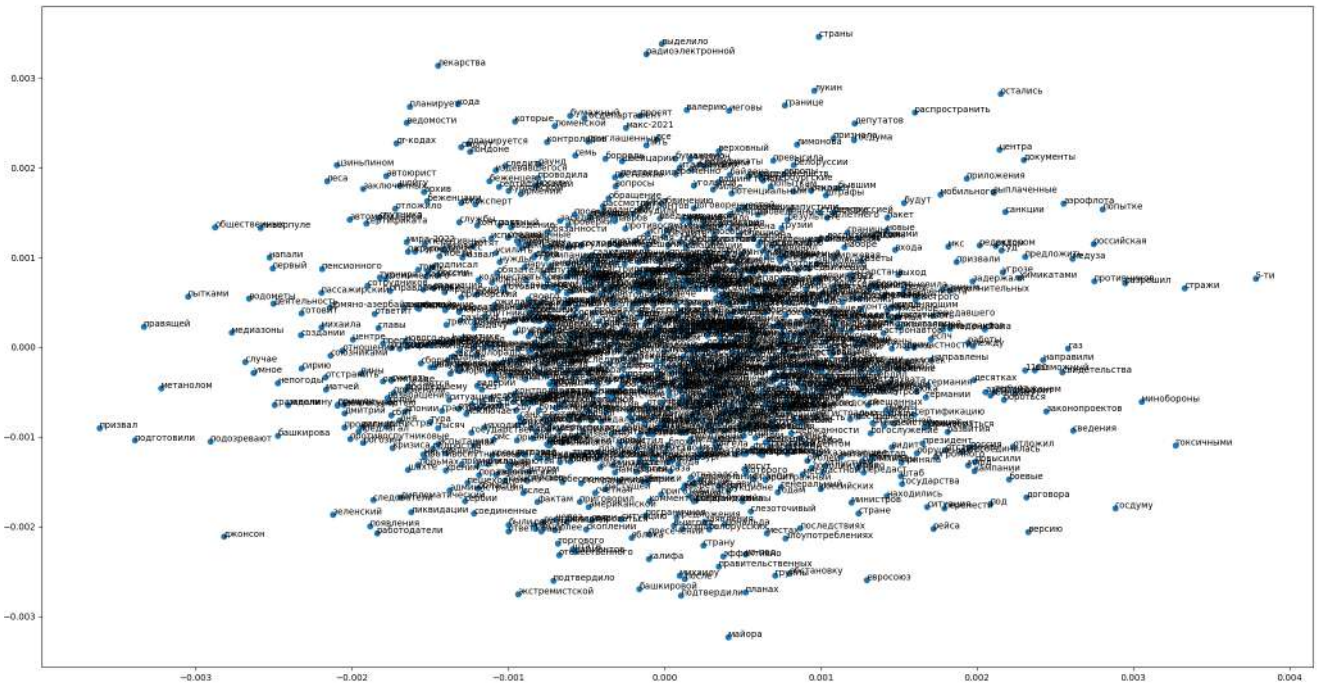


Рисунок 4.14 — Результат анализа новостных заголовков

Для реализации PCA использована библиотека *sklearn*. Результатом выполнения является представление слов из гипертекста в двухмерном пространстве. Визуализацию на плоскости можно сделать с помощью пакета *pyplot* из библиотеки *matplotlib*. Результат работы показан на рис. 4.14.

Видно, что построенная диаграмма не информативна, и ее сложно использовать для дальнейшего анализа. Поэтому по результатам проведенного онто-семантического анализа строится сеть, состоящая из набора графов. Вершины графа — слова, а ребра отражают наличие значимой связи между словами. Диаметр окружности вершины графа указывает на частоту конкретного слова в гипертексте, толщина линии ребра на силу связи.

Графы ежедневных актуальных новостных слов (кластеры) (см. рис. 4.15 и 4.16) задают динамику событий и позволяют формировать запросы для поиска соответствующих мемов. Еженедельно (или с другим периодом) полученные кластеры совместно формируют новый кластер с некоторым ядром слов. Маршрут между кластерами отражает связь между устойчивыми словосочетаниями (ежедневными) и выделяет граничные элементы кластеров. Аналогичный процесс происходит с мемами, соответствующими запросами по ежедневным кластерам актуальных словосочетаний. Для снижение размерности, выбора словосочетаний используется PCA.

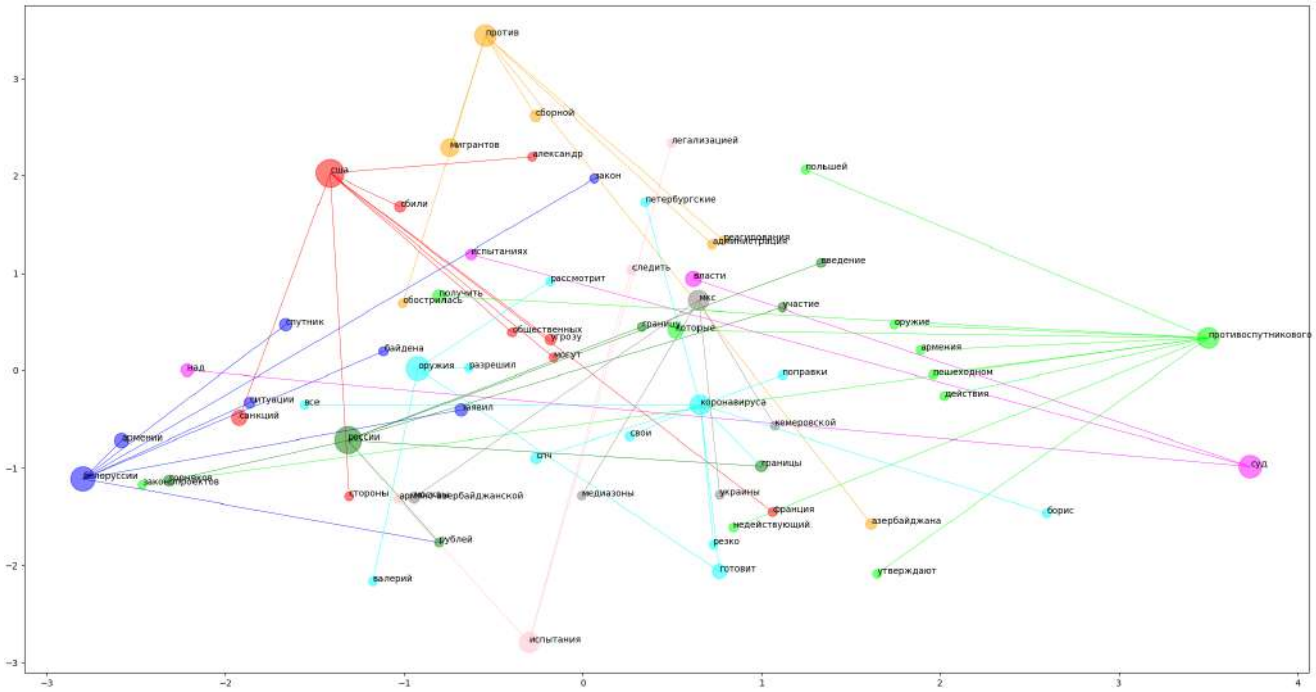


Рисунок 4.15 — Результат анализа новостных заголовков за 16.11.2021

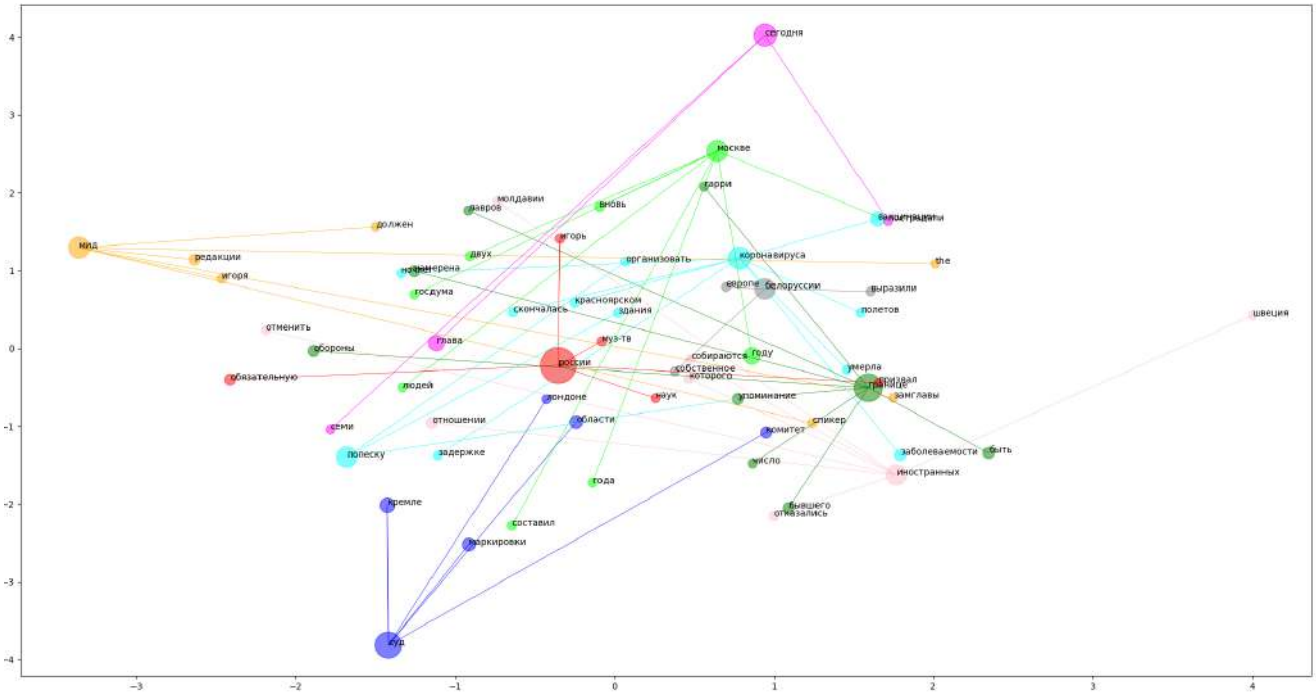


Рисунок 4.16 — Результат анализа новостных заголовков за 17.11.2021

На рис. 4.15 и 4.16 показаны результаты выполнения описанного алгоритма визуализации для полученной модели, построенной на основе новостных заголовков за 16.11.2021 и 17.11.2021. Как можно видеть из рисунков, что связанные друг с другом слова не всегда находятся рядом на плоскости. Такие особенности расположения на плоскости обоснованы тем, что в изначальной модели параметров значительно больше, чем мы можем показать на плоскости.

С точки зрения программной реализации структуру программного комплекса можно представить в виде рис. 4.17. Состоит из компонент:

— база данных — для хранения информации;

— файловое хранилище — для сохранения изображений (мемов);

— программа для сбора информации по поисковым запросам;

— программа анализа новостных заголовков и формирования запросов;

— сервис для определения дубликатов изображений;

— сервер, отвечающая за связь остальных компонентов.

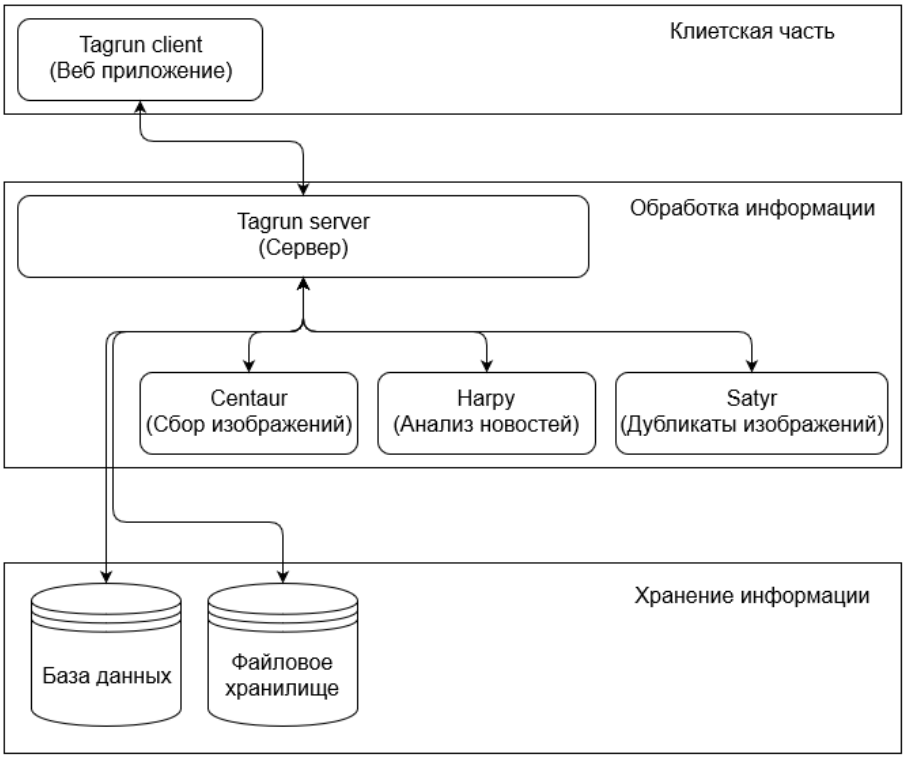


Рисунок 4.17 — Структура комплекса «Memometrix»

Для хранения информации об изображениях, мониторинговых запросах, результатов мониторинга, а также о тэгах и временных отметках используется реляционная база данных. Однако, база данных обеспечивает хранение только информации об объектах и их связях, но не обеспечивает хранение файлов изображений. Поэтому отдельно реализован сервис хранения для загружаемых файлов.

Для определения схожести изображений (нахождения дубликатов) необходимо особым образом хэшировать эти изображения. Эту задачу на себя берет сервис хэширования изображений. В данном случае под хэшированием понимается алгоритм создания хэшей с той особенностью, что хэши изображений с минимальными различиями должны быть максимально близки в определенной метрике.

Для наполнения базы данных актуальной информацией используется автоматический сборщик *Centaur*. Решение вынести логику сборки информации по поисковым запросам и

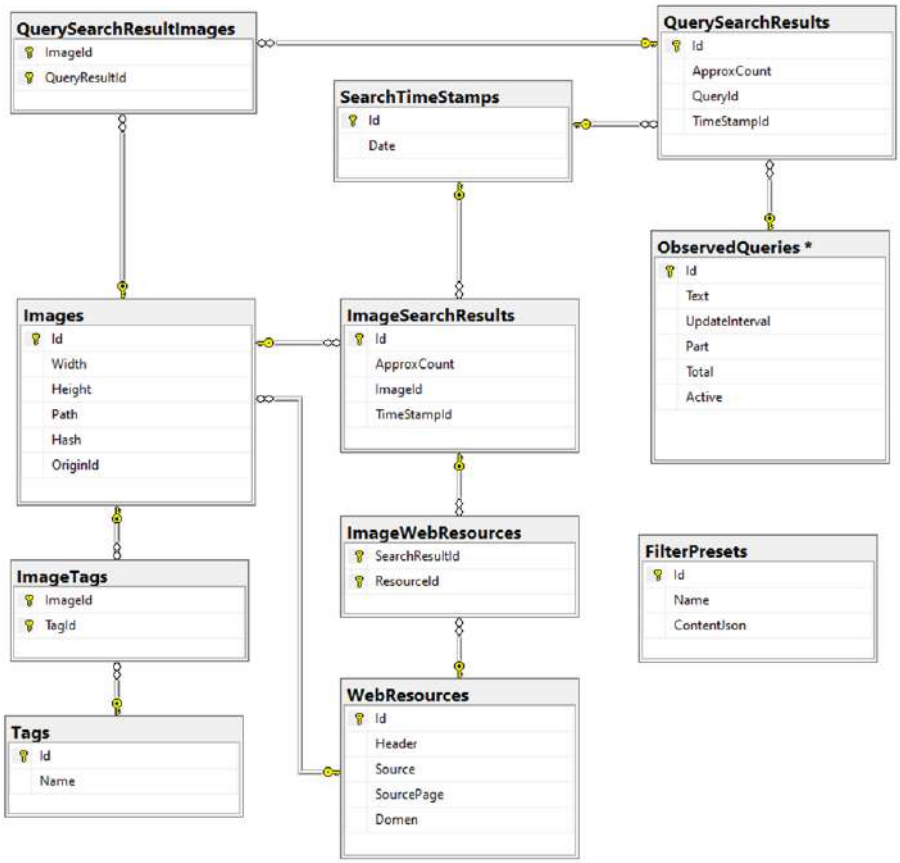


Рисунок 4.18 — Структура базы данных

обратному поиску изображений в отдельную программу связано с тем, что алгоритм сбора меняется в зависимости от ресурсов, с которых происходит сбор. Так как формат этих ресурсов не постоянный, то и программа может нуждаться в частых изменениях.

В базе данных (рис. 4.18) определены следующие сущности:

— *ObservedQueries* (мониторинговые запросы);

— *SearchTimeStamps* (даты поиска);

— *Images* (изображения);

— *Tags* (тэги);

— *WebResources* (ссылки на веб ресурсы);

— *QuerySearchResults* (результаты поиска по запросу);

— *ImageSearchResults* (результаты обратного поиска по изображению);

— *FilterPresets* (шаблоны фильтров).

Основным компонентом комплекса является *TagRun Server*, написанный на языке C# под платформу *.Net Core*. Он представляет собой связующий слой классической трехуровневой архитектуры. Главными фреймворками, использованными для его разработки, являются *ASP.NET* и *Entity Framework*.

Первый организует веб-сервер для приложения и дает возможность создавать обработчики для запросов. На основе *ASP.NET* реализован *REST API*, который используется остальными компонентами для взаимодействия. Логика работы с каждой из сущностей реализована в отдельном контроллере, а обмен данными происходит в формате *JSON*.

Клиентское приложение *TagRun Client* представляет собой веб-приложение на языке *JavaScript*. Использование технологий современной веб-разработки позволяет создать единое клиентское приложение для любых платформ, которые поддерживают веб браузер. Разработанный клиент взаимодействует с сервером по *REST*. Приложение состоит из двух экранов: работы с запросами и работы с фильтрами. Функционал клиента включает в себя:

— добавление, изменение и удаление запросов мониторинга;

— обзор собранных материалов;

— настройку фильтров полученных изображений;

— экспорт отфильтрованных изображений.

Главной библиотекой для создания веб приложения является *ReactJS*, который позволяет создавать компоненты интерфейса, которые автоматически обновляются при изменении состояния приложения (переход на другой экран, обновление содержимого и т. д.).

Сбором изображений занимается *Centaur*. Это приложение также написано на языке C#. Ежедневно оно получает информацию о запросах для мониторинга от сервера. По каждому из запросов осуществляется поиск в поисковой интернет системе (Google, Yandex), после чего собирает и сохраняет результаты обратно на сервер.

Роль вычисления дубликатов и похожих изображений берет на себя *Satyr*. Это приложение написанное на языке *Python* для определения схожести изображений. Для этого каждое изображение проходит алгоритм хэширование. В данном случае используется перцептивное хеширование. Его основное отличие от других алгоритмов в том, что данные проходят хэширование в качестве изображения, а не произвольной информации. Идея алгоритма состоит в том, что изображение сжимается до заданных размеров, затем приводиться к некоторому оттенку серого, также в зависимости от параметра (например при минимальном значении может быть только черный или белый цвет), после чего каждый из пикселей изображения выписывается в виде числового значения. Для сравнения изображений друг с другом достаточно высчитать разницу между соответствующими им хэшами, если она меньше определенного значения, то изображения с большой вероятностью похожи.

Harpy собирает информацию с новостных ресурсов, модуль написан на *Python*. После сбора информации о новостях происходит онто-семантический анализ с целью выделения общих тематик среди полученных данных. На основе выделенных групп слов формируются поисковые запросы, которые затем сохраняются на сервер.

В дальнейшем возможна доработка комплекса: анализ полученных результатов, визуализация собранной информации, прогнозирования дальнейшего распространения интернет-мемов.

На основании запросов «Memometrix» выдает ранжированный набор мемов и соответствующие им показатели по социальным сетям. На основании матрицы показателей строятся интегральные показатели влияния интернет-мемов на русскоязычных пользователей сети Интернет, а также индексы $I(t)$, $I^*(t)$, $t = \overline{1,T}$, отвечающие моментам получения данных.

Пусть $A = \{a_{ij}\}$, $\left(i = \overline{1,m},\ j = \overline{1,n}\right)$ — матрица показателей мемов.

Обозначим через

$$\alpha_j = \min_{1 \leqslant i \leqslant m} a_{ij}, \quad \beta_j = \max_{1 \leqslant i \leqslant m} a_{ij} \tag{4.1}$$

$$\alpha_j^* < \alpha_j, \quad \beta_j^* > \beta_j, \quad j = \overline{1,n}. \tag{4.2}$$

Предположим, что чем больше показатель a_{ij}, тем i-й мем распространяется и влияет больше.

Введем нормировку

$$x_{ij} = \frac{a_{ij} - \alpha_j}{\beta_j - \alpha_j}, \quad 0 \leqslant x_{ij} \leqslant 1, \quad i = \overline{1,m}, \quad j = \overline{1,n} \tag{4.3}$$

и

$$x_{ij}^* = \frac{a_{ij} - \alpha_j^*}{\beta_j^* - \alpha_j^*}, \quad 0 < x_{ij}^* < 1, \quad i = \overline{1,m}, \quad j = \overline{1,n}. \tag{4.4}$$

Коэффициенты α_j^*, β_j^* устанавливаются экспертно или уточняются в процессе установления обратной связи о безопасности (безобидности мемов) или их деструктивного характера. Границы безопасности могут соответствовать $\alpha_j^* > \alpha_j$ и $\beta_j^* < \beta_j$.

Нормировки (4.3), (4.4) являются монотонными и не искажают качественные показатели a_{ij}.

Найдем главные компоненты для множества показателей (нормированных) и выберем наиболее значимые (информационно):

$$y_{il}, \quad 1 \leqslant l \leqslant L << n. \tag{4.5}$$

Построим показатели

$$I_i = \sum_{l=1}^{L} \gamma_l \, y_{il}, \quad i = \overline{1,m}. \tag{4.6}$$

Не меняя значение коэффициентов, полученных при построении главных компонент y_{il} и γ_l в I_i, каждому мему будем ставить в соответствие набор показателей $I_i(t)$, $t = \overline{1,T}$, отвечающих моменту получения данных (например, каждый день).

Данный подход позволяет отслеживать жизненный цикл конкретного мема (или группы мемов). По графикам индекса I^* (нормированы показатели по границам безопасности) можно: выяснять их безопасность; отсекать от рассмотрения мемы с незначительными значениями показателей (индексов) I, I^*; проводить кластеризацию мемов; проводить анализ запросов на выявление новых мемов и их распространение.

Подробное описание системы интеллектуализированного анализа программного комплекса «Memometrix» и возможности анализа, мониторинга распространения и влияния интернет-мемов приведено в совместной работе [76].

4.3.2 Некоторые задачи обработки интернет-мемов

Рассмотрим задачи обработки интернет-мемов связанные со структурами графов: выделение сообществ (кластеризация) и связи элементов графовых структур (маршрутизация). В

работе автора и соавторов (М. Г. Козловой, В. А. Лукьяненко [33]) подробно рассматривается специфика обработки информации потока интернет-мемов с позиции сложных сетей.

Рассмотрим более подробно разработку кластеризации социальных сетей относительно популярности тематик.

Задача кластеризации сообщества социальной сети «*ВКонтакте*». Для реализации алгоритмов в качестве языка программирования выбран *Python* с использованием среды программирования *PyCharm* и *Jupyter Notebook*. Будем использовать данные, полученные из социальной сети «*ВКонтакте*». Для сбора информации из сети в пакетах *Python* реализована разработанная «обертка» с помощью модуля *request* и *VKAPI*, с использованием общественного уровня доступа сети «*ВКонтакте*», который не разрешает использовать информацию о сообщениях или о скрытой информации пользователя, но разрешает использовать данные открытых групп, основную информацию о пользователях и их подписках. То есть используется существующая информационная модель сети и интеллектуализация обработки данных сети.

Основной метод определения популярных тематик состоит в первичном анализе данных, по результатам которого будет выполнятся кластеризация.

Для анализа данных используется мера *TFIDF* (англ. *Term frequency* — частота терма, англ. *Inverse document frequency* — обратная частота документа). Она оценивает важность слова внутри конкретного документа, среди набора разного рода других документов. Идея заключается в том, что слова, которые часто употребляются в одном конкретном документе, но редко в остальных, будут иметь наибольший вес. Действительно, если слово упоминается пару раз во всех документах — это значит, что слово общеупотребительное и семантической важности не несет. Если же слово упоминается часто лишь в одном документе — значит, что этот документ, скорей всего, тематически связан с данным словом. Соответственно, редко используемые слова в документах — будут представлять большинство, и их можно обрезать с помощью параметров алгоритма [219].

Мера *TFIDF* часто используется в задачах анализа текста, поскольку соответствующий алгоритм легок, быстр в реализации и выполнении, а также эффективен в реальных задачах, например, в построении поисковых запросов.

Алгоритм состоит из двух основных частей:

1. TF — частота терма. Оценивает важность слова внутри одного документа. Пусть w — слово, важность которого необходимо определить, тогда мера TF вычисляется следующим образом:

$$TF\,(w,\,d) = \frac{n_w}{n_d},$$

где n_w — число вхождений слова w в документ d, а n_d — общее число слов в d. В качестве меры можно взять и булеан, тогда TF будет высчитываться следующим образом:

$$TF\,(w,d) = \begin{cases} 1, \text{если } w \text{ есть в } d, \\ 0, \text{ иначе.} \end{cases}$$

2. IDF — обратная частота появления слова в документах. Она определяет то, сколько информации содержится в слове, насколько часто оно употребляется в других документах.

Для каждого слова среди всего множества документов оценка IDF определяется однозначно [200]:

$$DF\,(w,D) = log\frac{|D|}{|d \in D, w \in d|}.$$

Основание логарифма принято брать равное 2 или 10.

Соответственно, мера $TFIDF$ является произведением двух мер TF и IDF:

$$TFIDF(w,d,D) = TF(w,d) \cdot IDF(w,D).$$

Мера $TFIDF$ применяется не только к словам, но и, например, к ссылкам. Если одна ссылка присутствует в нескольких документах, то она должна получить больший вес, чем ссылки, которые присутствуют в большом количестве документов.

Есть также вопрос важности компоненты IDF при использовании $TFIDF$. Многочисленные исследования показали, что, добавляя к мере TF меру IDF, исследователи не заметили существенной разницы в эффективности и адекватности результата [219].

Также в качестве семантического анализа применялся алгоритм *LDA* (англ. *Latent Dirichlet Allocation* — латентное размещение Дирихле) — модель, которая позволяет определить самые релевантные темы из данных. Идея состоит в том, что слова собраны в документ, который представляет собой смесь различных тем, и наличие каждого слова в документе непосредственно связанно с одной из тем документа. Он был впервые представлен Дэвидом Блеем, Эндрю Ыном и Майклом Джорданом в 2003 году [160].

Метод *LDA* основан на вероятностном латентном семантическом анализе [15]. Вероятностную модель появления пары «слово-документ» (w,d) можно записать следующим образом:

$$p\,(w,d) = \sum_{t \in T} p\,(d)\,p(w|t)p(t|d),$$

где T — множество тем, $p\,(t)$ — неизвестное априорное распределение тем по всем документам; $p\,(d)$ — априорное распределение на множестве всех документов, $p\,(d) = \frac{n_d}{n}$, $n = \sum\limits_d n_d$ — эмпирическая оценка, n — количество слов во всех документах; $p\,(w)$ — априорное распределение на множестве слов, аналогично, $p\,(w) = \frac{n_w}{n}$, n_w — эмпирическая оценка, n_w — число вхождений слова w во все документы; условные вероятности $p(w|t)$ и $p(t|d)$ выражаются по формуле Байесса.

Анализ полученных данных. У каждого пользователя в профиле имеется поле «Интересы», однако оно является свободным для заполнения, отсюда следует что информативность и адекватность данного поля мала, учитывая, что оно может быть пустым, и среднее количество пользователей, заполняющих данное поле, в среднем ≈ 10 % от общего числа людей в группе.

Более информативной является информация о подписках пользователя на различные группы. Все группы упорядочены по частоте посещения пользователем — чем чаще посещает, тем выше в списке, следовательно, самые первые группы и будут представлять наибольший интерес пользователя. Тематика группы определяется из 3-х полей: название, описание и

категория группы. Последнее — самое полезное, однако, присутствует не у всех групп. Эти «дыры» заполнятся тематикой, собранной из описаний или названий. Для этого сначала применяется нормализация текста, т. е. приводится текст в нормальную форму, и прогонка текста методом семантического / частотного анализа. Далее, с помощью словаря, находится ближайшее совпадение полученных тематик с существующими категориями и определяется соответствующий интерес. Для нормализации текста подойдет любой лемматизатор, то есть приведение словоформы к лемме — ее нормальной (словарной) форме. Был использован модуль *pymorphy2* [99] в совокупности с $TFIDF$ векторизатором.

Чтобы убрать тривиальные тематики (например, «Юмор», поскольку большое число людей подписаны на группы юмористического характера) или явные выбросы, был введен список стоп-слов, а также создан словарь для группировок. Так, «СМИ», «Интернет-СМИ» и «Информационный портал» будут сгруппированы в «СМИ», а «Футбол», «Футбольный клуб» и «Футбольный стадион», после нормализации текста будут сгруппированы в «Футбол».

В качестве примера, рассмотрим группу математического факультета КФУ им. В. И. Вернадского (выборка ≈ 1500 пользователей).

Учитывая взятую группу, и тот факт, что большинство пользователей этой группы — студенты, такие интересы ожидаемы. Рассмотрим информацию пользователей по интересам. Для этого каждому пользователю в соответствие сопоставим вес интереса, посчитанный по формуле:

$$weight(interest)^u = \frac{\sum\limits_{j \in interest} group_{u,j}}{\sum\limits_{i=0}^{n} group_{u,i}},$$

где $weight$ — вес интереса, u — текущий пользователь, $interest$ — текущий уникальный интерес пользователя, $j \in interest$ — индексы групп, которые соответствуют данному интересу, $group$ — список групп пользователя, n — количество групп пользователя, $group_{u,i}$ i-я группа пользователя u.

В итоге получим для каждого пользователя набор интересов с некоторым весом важности, например: 0.25 — наука, 0.2 — фотография, 0.2 — программирование.

Представим все интересы и связи в виде сети (графа). Каждый пользователь будет однозначно определен к одному интересу (интерес с наибольшим весом), а связи между пользователями будут формироваться из оставшихся интересов, в порядке их важности. Количество связей между пользователями и количество интересов одного пользователя, которое следует учитывать в алгоритме — настраивается и дает разные изображения. Для визуализации графа использовался модуль $Networkx$. Результаты работы представлены ниже.

Из рис. 4.19, 4.20 выделим «Образование», «Общественная деятельность», «Кино» и «СМИ», которые представлены наибольшими по размеру кластерами. Заметим близость между кластерами «Общественная деятельность» и «Кино», свидетельствующая о том, что людей, которых интересует «Кино» также сильно интересует и «Общественная деятельность», поскольку они имеют высокую степень связности друг с другом. Можно выделить

кластеры, расположенные снизу: «Домашние и дикие животные», «Культура», «Астрология», «Одежда и мода», «Родители и дети», «Фитнес», «Молодежное движение» и «Уход за собой», которые расположены рядом и семантически сильно связаны женским полом.

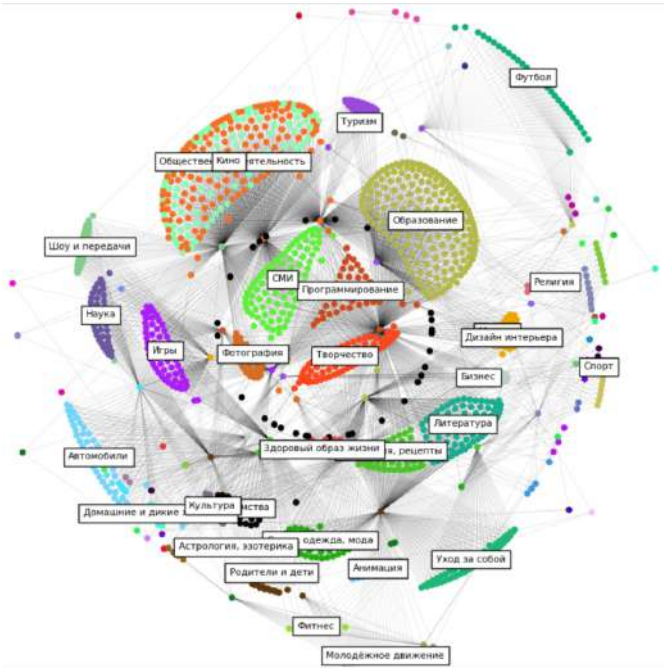


Рисунок 4.19 — Сеть пользователей группы факультета математики и информатики «КФУ им. В. И. Вернадского», кластеризованных по интересам. Каждая вершина — это пользователь, где учитываются первые 3 интереса по весу, максимальное количество связей между вершинами — 3, текстом обозначены кластеры, имеющие более 5 пользователей.

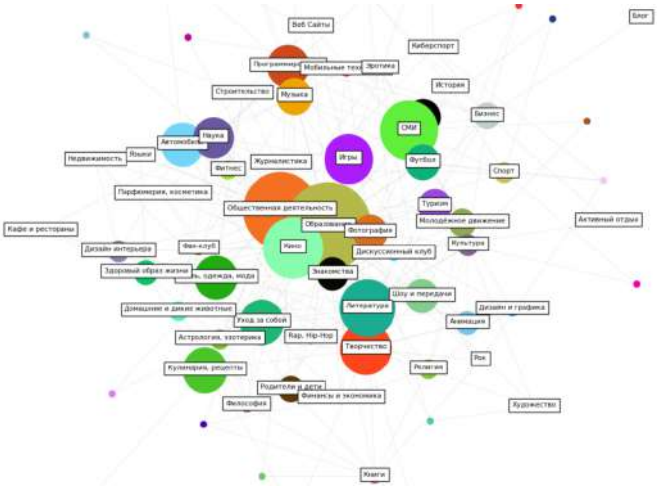


Рисунок 4.20 — Сеть пользователей группы факультета математики и информатики «КФУ им. В. И. Вернадского», кластеризированных по интересам. Каждая вершина — кластер, где размер соответствует количеству пользователей в кластере, количество связей между кластерами — не более 3.

Выводы

1. Прикладные задачи маршрутизации в сложных сетях в условиях чрезвычайных ситуаций приводят к новым постановкам задач псевдобулевой (дискретной) оптимизации с различными ограничениями.

2. Разработанные алгоритмы маршрутизации по доставке водных ресурсов и сбалансированный маршрут $mTSP$ реализованы в программном комплексе «Программа многоагентной инфраструктурной маршрутизации». Приведено описание работы комплекса, архитектуры и алгоритмического наполнения, а также визуализация $mTSP$ на реальных картах.

3. В программном комплексе «Программа выбора наилучших туристических маршрутов по Крыму» на основе картографических сервисов реализован выбор маршрутов по достопримечательностям с удобным для пользователя интерфейсом.

4. Разработан программный комплекс «Memometrix», в котором реализован алгоритм кластеризации сообществ социальных сетей с целью выявления групп, подверженных влиянию.

Заключение

Основные результаты работы заключаются в следующем.

1. Выделен класс постановок модельных задач TSP, $mTSP$ и полиномиальных алгоритмов их решения, пригодных для синтеза комбинированных алгоритмов, в которых учитываются: прикладной характер моделей, знания о модели и сложной структуре сети, прецедентные знания и возможность реоптимизации. Представлены исторические аспекты по TSP, их обобщениям, точным и приближенным алгоритмам решения.

В прикладной теории графов, предназначенной для решения различных задач прокладки маршрутов (замкнутых, разомкнутых, кратчайших, критических и т. п.) в сложных сетях возникает ряд ограничений, условий, предписаний: как в стационарном случае, когда задача решается заранее и при всех известных условиях, так и нестационарном, когда информация появляется в процессе прокладки маршрута. Такая информация может служить для формализации модельных ситуаций и быть наполнением для программных агентов по локальному поведению для достижения глобальной цели или группы, решающих общую задачу.

2. Обосновано представление $mTSP$ как модели псевдобулевой оптимизации, в которой часть ограничений (или все) могут быть заданы в виде $ДНФ$ ограничений. При этом процедура поиска и логического вывода о принадлежности к искомому решению является полиномиальной.

Показано, что модели в виде псевдобулевой оптимизации с $ДНФ$ ограничениями служат теоретической основой для MAC типа $mTSP$ и систем управления. Рассмотрены алгоритмы декомпозиции, кластеризации, анализа и синтеза сети в задачах типа многих агентов-коммивояжеров.

3. Разработана процедура снижения размерности исходной задачи $mTSP$ с помощью кластеризации сложных сетевых структур и итерационного уточнения кластеров в зависимости от решения TSP на каждом кластере и в целом.

Показано, что методология разработки алгоритмов решения задач маршрутизации может быть основана на формировании по исходной сложной сети более простой по своей структуре сети (относительно реализации алгоритмов маршрутизации).

Обоснована перспективность метода кластеризации в сочетании с метаэвристиками. В предположении, что проведена предварительная обработка сети с учетом ее структуры, т. е. найдено разбиение графа сети на подграфы (кластеризация), задача построения маршрутов коммивояжера решается на сети меньшей размерности. Сравнивается работа алгоритмов муравьиной колонии, имитации отжига, пчелиной колонии. Результаты по гибридным алгоритмам показывают возможность оптимизационной комбинации базовых алгоритмов.

4. Проведены численная реализация алгоритмов (точных, эвристик, метаэвристик) и вычислительные эксперименты по кластеризации (максимальный разрез и другие) для построения решений $mTSP$. Разработана программная реализация алгоритмов кластеризации: иерархический алгоритм, $K\text{-}means$ и жадный алгоритм с различными модификациями. Реализован генетический алгоритм для решения TSP, а также синтез алгоритмов кластеризации

и решения *TSP* с нахождением оптимальных центров. Обмен информацией между агентами реализован в виде механизма «перебрасывания» вершин из более крупных кластеров в более мелкие.

При наличии статистики распределения весов (дуг) сети эффективным является использование процедур максимального разреза. Найденные маршруты могут повторно использоваться, т. е. адаптироваться к изменяющимся условиям с целью получения маршрутов полиномиальной сложности. На базе реализованных алгоритмов проведен вычислительный эксперимент, результаты которого подтверждают практическую пригодность принятого подхода.

5. Разработаны программные комплексы: «Программа многоагентной инфраструктурной маршрутизации», «Программа выбора наилучших туристических маршрутов по Крыму», «Программный комплекс Memometrix для исследования влияния политических мемов на пользователей Рунета».

В заключение автор выражает благодарность и большую признательность научному руководителю М. Г. Козловой за поддержку, помощь и научное руководство, а также автор благодарит В. А. Лукьяненко за помощь в работе и обсуждении результатов, оформлении диссертации.

Список сокращений и условных обозначений

2-*Opt*, 3-*Opt*, 5-*Opt* — алгоритмы локального поиска

ACO — алгоритм муравьиной колонии

AC2OptGA — алгоритм преобразует найденное решение TSP в решение $mTSP$

AmTSP — алгоритм решения задачи для двух коммивояжеров

ES — алгоритм эволюционных стратегий

GA — генетический алгоритм

GA-ACO — гибридный алгоритм, состоящий из *GA* и *ACO*

GA2OPT — гибридный метаэвристический алгоритм локального поиска гравитационной эмуляции

GES — метод глобального равновесного поиска

GIACO — гибридный алгоритм, состоящий из *GA* и интеллектуальной и тупой оптимизации колоний муравьев (*IDACO*)

GP — алгоритм генетического программирования

GTSP — гамильтонова задача коммивояжера

K-means — алгоритм кластеризации

LBA — алгоритм баланса нагрузки

MACO — модифицированный алгоритм колонии муравьев

MAX-CUT — максимальный разрез

MGA — модифицированный генетический алгоритм

M-GELS — модифицированный алгоритм локального поиска гравитационной эмуляции

$mTSP$ — многоагентная задача маршрутизации

PSO — метод роя частиц

PR — метод связывающих путей

QA — алгоритм для асимметричных $mTSP$

RandIns — случайная вставка

SEC — алгоритм исключения субтура

SW + ASelite — комбинация алгоритма *Sweep*, алгоритма элитной колонии муравьев и 3-*Opt*

Sweep — алгоритм развертки

TSA — поисковый алгоритм с запретами

TSPLIB — это библиотека примеров экземпляров *TSP* (и связанных с ними проблем) из различных источников и типов

TSP — задача коммивояжера или Traveling Salesman Problem

БЗ — база знаний

БПЛА — беспилотный летательный аппарат

ДНФ — дизъюнктивная нормальная форма

ДО — дискретная оптимизация

ЗДО — задача дискретной оптимизации

ЗКО — задача комбинаторной оптимизации

ЗЛП — задача линейного программирования

ИА — интеллектуальный агент

ИИ — искусственный интеллект

ИНС — искусственная нейронная сеть

ИС — интеллектуальная система

ИСУ — интеллектуальная система управления

КНФ — конъюнктивная нормальная форма

ЛЗН — линейная задача о назначениях

ЛПР — лицо, принимающее решение

ЛСП — логическая система продукций

МА — муравьиный алгоритм

МАИС — многоагентная интеллектуальная система

МАС — многоагентная система

МРЧ — метод роящихся частиц

ССЗ — стратегическая ситуационная зона

СУ — системы управления

ФАЛ — функция алгебры логики

ЧС — чрезвычайная ситуация

ЭГА — эволюционно-генетический алгоритм

Список литературы

1. Алгоритм для решения задачи о коммивояжере / Д. Литл [и др.] // Экономика и матем. методы. — 1965. — Т. 1, № 1. — С. 94—107.
2. Алгоритмы и методы решения задач составления расписаний и других экстремальных задач на графах больших размерностей / Е. В. Панкратьев [и др.] // Фундамент. и прикл. матем. — 2003. — Т. 9, № 1. — С. 235—251.
3. Алгоритмы: построение и анализ / Т. Кормен [и др.]. — 3-е изд. : Пер. с англ. — М. : ООО «И. Д. Вильямс», 2013. — 1328 с.
4. *Алексеев, В. Е.* Графы. Модели вычислений. Структуры данных / В. Е. Алексеев, В. А. Таланов. — Нижний Новгород : Изд-во ННГУ, 2005. — 307 с.
5. *Алексеева, Е. В.* Построение математических моделей целочисленного линейного программирования. Примеры и задачи : Учеб. пособие / Е. В. Алексеева. — Новосибирск : Новосиб. гос. ун-т, 2012. — 131 с.
6. *Антамошкин, А. А.* Поисковые алгоритмы псевдобулевой оптимизации / А. А. Антамошкин, И. С. Масич // Системы управления, связи и безопасности. — 2016. — № 1. — С. 103—145.
7. *Артюхин, В. В.* Прогнозирование чрезвычайных ситуаций с помощью дискретной оптимизации и современных программных средств / В. В. Артюхин // Технологии гражданской безопасности. — 2014. — Т. 11, № 1. — С. 86—91.
8. *Беллман, Р.* Динамическое программирование / Р. Беллман. — М. : Иностранная литература, 1960. — 400 с.
9. *Берцун, В. Н.* Математическое моделирование на графах. Часть 2 / В. Н. Берцун. — Томск : Изд-во Томск. ун-та, 2013. — 88 с.
10. *Быкова, В. В.* Структурная декомпозиция графа и ее применение для решения оптимизационных задач на разреженных графах / В. В. Быкова // Прикладная дискретная математика. Приложение. — 2014. — № 7. — С. 154—157.
11. *Быкова, В. В.* Адаптивное размещение ориентиров в задаче о кратчайшем пути для графа большой размерности / В. В. Быкова, А. А. Солдатенко // Программные продукты и системы. — 2016. — № 1. — С. 60—67.
12. *Васильев, О. В.* Модифицированный алгоритм муравьиных колоний для решения задачи коммивояжера / О. В. Васильев, Т. М. Леденева // Системы управления и информационные технологии. — 2011. — Т. 45, № 3.2. — С. 293—297.
13. *Виноградов, А. Н.* Динамические интеллектуальные системы. Ч. 1. Представление знаний и основные алгоритмы / А. Н. Виноградов, Г. С. Осипов, Л. Ю. Жилякова // Известия РАН. Теория и системы управления. — 2002. — № 6. — С. 119—127.
14. *Виноградов, А. Н.* Динамические интеллектуальные системы. Ч. 2. Моделирование целенаправленного поведения / А. Н. Виноградов, Г. С. Осипов, Л. Ю. Жилякова // Известия РАН. Теория и системы управления. — 2003. — № 1. — С. 87—94.

15. *Воронцов, К. В.* Вероятностное тематическое моделирование: теория, модели, алгоритмы и проект BigARTM / К. В. Воронцов. — 2021. — URL: http://www.machinelearning.ru/wiki/images/d/d5/Voron17survey-artm.pdf (дата обр. 01.03.2021).
16. *Воронцов, К. В.* Лекции по алгоритмам кластеризации и многомерного шкалирования / К. В. Воронцов. — URL: http://www.ccas.ru/voron/download/Clustering.pdf (дата обр. 01.03.2021).
17. *Гаращенко, И. В.* Полиномиальное преобразование в приближенных алгоритмах решения задач типа коммивояжера / И. В. Гаращенко, А. В. Панишев, О. Б. Маций // Радиоэлектроника и информатика. — 2007. — № 1. — С. 45—49.
18. *Германчук, М. С.* Использование дополнительной информации в задачах дискретной оптимизации типа многих коммивояжеров / М. С. Германчук // Таврический вестник информатики и математики. — 2016. — Т. 33, № 4. — С. 68—82.
19. *Германчук, М. С.* Использование дополнительной информации в знаниеориентированных задачах типа коммивояжера / М. С. Германчук // XXVI Крымская Осенняя Математическая Школа-симпозиум по спектральным и эволюционным задачам (КРОМШ-2015): сборник тезисов. — 2015.
20. *Германчук, М. С.* Многоагентный подход в сетевых задачах / М. С. Германчук // Международная конференция «XXVII Крымская Осенняя Математическая Школа–симпозиум по спектральным и эволюционным задачам» (КРОМШ–2016): сборник тезисов. — 2016.
21. *Германчук, М. С.* Многоагентный подход к задачам типа многих коммивояжеров на сложных сетях / М. С. Германчук // Дистанционные образовательные технологии: материалы III Всероссийской научно-практической конференции (г. Ялта, 17-22 сентября 2018 года) / отв. ред. В. Н. Таран. — Симферополь : ИТ «АРИАЛ», 2018. — С. 208—217.
22. *Германчук, М. С.* Разрешимость задач псевдобулевой условной оптимизации типа многих коммивояжеров / М. С. Германчук // Таврический вестник информатики и математики. — 2020. — Т. 49, № 4. — С. 30—55.
23. *Германчук, М. С.* Система интеллектуального управления в прикладных сетевых задачах / М. С. Германчук // Научно-практическая конференция «Молодая наука»: сборник трудов / под общей редакцией Н. Г. Гончаровой. — Симферополь : ИТ «АРИАЛ», 2015. — С. 46—48.
24. *Германчук, М. С.* Сравнение математических моделей и выбор алгоритмов в задачах маршрутизации / М. С. Германчук, Д. А. Игнатенко // Математика, информатика, компьютерные науки, моделирование, образование: сборник научных трудов научно-практической конференции МИКМО-2018 и Таврической научной конференции студентов и молодых специалистов по математике и информатике / Под ред. В. А. Лукьяненко. — Симферополь : ИП Корниенко А. А., 2018, Вып. 1. — С. 149—154.
25. *Германчук, М. С.* Задача реоптимизации сети / М. С. Германчук, М. Г. Козлова // Математика, информатика, компьютерные науки, моделирование, образование: сб. науч. трудов научно-практической конференции МИКМО-2017 и Таврической научной

конференции студентов и молодых специалистов по математике и информатике / Под ред. В. А. Лукьяненко. — Симферополь : ИП Корниенко А. А., 2017. — С. 109—113.

26. *Германчук, М. С.* Использование информации в задачах типа многих коммивояжеров / М. С. Германчук, М. Г. Козлова // XXV Крымская Осенняя Математическая Школа-симпозиум по спектральным и эволюционным задачам (КРОМШ-2014). Тез. докладов. — 2014.
27. *Германчук, М. С.* Разработка алгоритмов кластеризации сложных (социальных) сетей / М. С. Германчук, М. Г. Козлова // Анализ, моделирование, управление, развитие социально-экономических систем: сборник научных трудов XIII Международной школы-симпозиума АМУР-2019, Симферополь-Судак, 14-27 сентября 2019 / Под общей редакцией А. В. Сигала. — Симферополь : ИП Корниенко А. А., 2019. — С. 118—126.
28. *Германчук, М. С.* Синтез алгоритмов кластеризации для решения многоагентной задачи коммивояжера / М. С. Германчук, М. Г. Козлова // Таврический вестник информатики и математики. — 2018. — Т. 39, № 2. — С. 49—70.
29. *Германчук, М. С.* Управление в многоагентных системах / М. С. Германчук, М. Г. Козлова // Международная конференция «Метод функций Ляпунова и его приложения»: тез. докл.; 15–18 сентября 2016 г. — Симферополь : Крымский федеральный ун-т им. В. И. Вернадского; отв. ред. О. В. Анашкин, 2016. — С. 46—47.
30. *Германчук, М. С.* Задачи маршрутизации в чрезвычайных условиях / М. С. Германчук, М. Г. Козлова, В. А. Лукьяненко // Анализ, моделирование, управление, развитие социально-экономических систем: сборник научных трудов XIV Всероссийской с международным участием школы-симпозиума АМУР-2020, Симферополь-Судак, 14-27 сентября 2020 / ред. совет: А. В. Сигал (предс.) и др. — Симферополь : ИП Корниенко А. А., 2020. — С. 98—107.
31. *Германчук, М. С.* Задачи практической маршрутизации / М. С. Германчук, М. Г. Козлова, В. А. Лукьяненко // Анализ, моделирование, управление, развитие социально-экономических систем. Сборник научных трудов XI Международной школы-симпозиума АМУР-2017. — 2017. — С. 116—120.
32. *Германчук, М. С.* Знаниеориентированные модели маршрутизации многих коммивояжеров / М. С. Германчук, М. Г. Козлова, В. А. Лукьяненко // Интеллектуализация обработки информации: Тезисы докладов 13-й Международной конференции, г. Москва. — 2020.
33. *Германчук, М. С.* Программные инструменты и технологии анализа потока интернет-мемов / М. С. Германчук, М. Г. Козлова, В. А. Лукьяненко // Таврический вестник информатики и математики. — 2020. — Т. 48, № 3. — С. 37—58.
34. *Германчук, М. С.* Псевдобулевые модели условной оптимизации для класса задач многих коммивояжеров / М. С. Германчук, М. Г. Козлова, В. А. Лукьяненко // Автомат. и телемех. — 2021. — № 10. — С. 25—45.
35. *Германчук, М. С.* Многоагентный подход к решению задачи коммивояжера / М. С. Германчук, М. Г. Козлова, А. Е. Пивовар // Математика, информатика, компьютерные

науки, моделирование, образование: сб. науч. трудов научно-практической конференции МИКМО-2017 и Таврической научной конференции студентов и молодых специалистов по математике и информатике / Под ред. В. А. Лукьяненко. — Симферополь : ИП Корниенко А. А., 2017. — С. 114—119.

36. *Германчук, М. С.* Метаэвристические алгоритмы для многоагентных задач маршрутизации / М. С. Германчук, Д. В. Лемтюжникова, В. А. Лукьяненко // Проблемы управления. — 2020. — Т. 6. — С. 3—13.
37. *Германчук, М. С.* Построение характеристик усреднения функций на сетевых структурах символического образа динамических систем / М. С. Германчук, В. А. Лукьяненко, О. С. Кортнева // «Информационные системы и технологии в моделировании и управлении»: сборник трудов V Международной научно-практической конференции (20–22 мая 2020 г.) / отв. редактор К. А. Маковейчук. — Симферополь : ООО «Издательство типография “Ариал”», 2020. — С. 186—195.
38. *Германчук, М. С.* Задача распознавания символического образа динамической системы / М. С. Германчук, В. А. Лукьяненко, А. О. Меньшиков // Математические методы распознавания образов: Тезисы докладов 19-й Всероссийской конференции с международным участием, г. Москва. — 2019.
39. *Гордеев, Э. Н.* Общий подход к исследованию устойчивости решений в задачах дискретной оптимизации / Э. Н. Гордеев, В. К. Леонтьев // Журнал вычислительной математики и математической физики. — 1996. — Т. 36, № 1. — С. 66—72.
40. *Гордеев, Э. Н.* Устойчивость в задачах на узкие места / Э. Н. Гордеев, В. К. Леонтьев // Журнал вычислительной математики и математической физики. — 1980. — Т. 20, № 4. — С. 1071—1075.
41. Графы с нестандартной достижимостью. Задачи, приложения: моногр. / Я. М. Ерусалимский [и др.]. — Ростов-на-Дону : ЮФУ, 2009. — 195 с.
42. *Гэри, М. М.* Вычислительные машины и труднорешаемые задачи / М. М. Гэри, Д. Д. Джонсон. — М. : Мир, 1982. — 416 с.
43. *Данильченко, М. Н.* Нейросетевой подход к построению маршрута в автоматизированной системе управления специального назначения / М. Н. Данильченко, А. Б. Муравник // Наукоемкие технологии в космических исследованиях Земли. — 2021. — Т. 13, № 1. — С. 58—66.
44. *Девятерикова, М. В.* Анализ устойчивости l-разбиения множеств в конечномерном пространстве / М. В. Девятерикова, А. А. Колоколов // Дискретн. анализ и исслед. опер. — 2000. — 7:2. — С. 47—53.
45. *Девятерикова, М. В.* Анализ устойчивости некоторых алгоритмов дискретной оптимизации / М. В. Девятерикова, А. А. Колоколов // Автоматика и телемеханика. — 2004. — № 3. — С. 48—54.
46. *Девятерикова, М. В.* Об устойчивости некоторых алгоритмов целочисленного программирования / М. В. Девятерикова, А. А. Колоколов // Известия вузов. Математика. — 2003. — № 12. — С. 41—48.

47. *Демиденко, В. М.* Специальный случай задачи о бродячем торговце / В. М. Демиденко // Весці. акад. навук Беларус. ССР. — 1976. — № 5. — С. 28—32.
48. *Демиденко, В. М.* Условия полиномиальной разрешимости задачи о коммивояжере и верхние оценки ее оптимума / В. М. Демиденко, В. С. Гордон, Ж.-М. Прот // Докл. НАН Беларуси. — 2003. — Т. 47, № 1. — С. 36—40.
49. *Донец, Г. А.* Метод моделирования структуры исходных данных и подклассы разрешимых задач комбинаторной оптимизации / Г. А. Донец, И. В. Сергиенко // Кибернетика и системный анализ. — 2014. — Т. 50, № 1. — С. 3—10.
50. *Донской, В. И.* Задачи псевдобулевой оптимизации с дизъюнктивным ограничением / В. И. Донской // Журнал выч. математики и матем. физики. — 1994. — № 4. — С. 461—472.
51. *Донской, В. И.* Интеллектуальное управление: обзор / В. И. Донской // Таврический вестник информатики и математики. — 2014. — № 2. — С. 14—35.
52. *Донской, В. И.* Дискретные модели принятия решений при неполной информации / В. И. Донской, А. И. Башта. — Симферополь : Таврия, 1992. — 166 с.
53. *Емеличев, В. А.* Анализ устойчивости эффективного решения век торной задачи о максимальном разрезе графа / В. А. Емеличев, К. Г. Кузьмин // Дискретн. анализ и исслед. опер. — 2013. — Т. 20, № 4. — С. 27—35.
54. *Емеличев, В. А.* Устойчивость в векторных комбинаторных задачах оптимизации / В. А. Емеличев, К. Г. Кузьмин, А. М. Леонович // Автоматика и телемеханика. — 2004. — № 2. — С. 79—92.
55. *Емеличев, В. А.* О количественной мере устойчивости векторной задачи целочисленного программирования / В. А. Емеличев, Д. П. Подкопаев // Журнал вычислительной математики и математической физики. — 1998. — Т. 38, № 11. — С. 1801—1805.
56. *Ерзин, А. И.* Задачи маршрутизации. Учебное пособие / А. И. Ерзин, Ю. А. Кочетов. — Новосибирск : Редакционно-издательский центр НГУ, 2014. — 96 с.
57. *Жилякова, Л. Ю.* Теория ресурсных сетей: монография / Л. Ю. Жилякова, О. П. Кузнецов. — М. : РИОР.ИНФА-М, 2017. — 283 с.
58. *Жукова, Г. Н.* Эффективный по времени точный комбинированный алгоритм для асимметричной задачи коммивояжера / Г. Н. Жукова, М. В. Ульянов, М. И. Фомичев // Бизнес-информатика. — 2018. — Т. 45, № 3. — С. 20—28.
59. *Журавлев, Ю. И.* О локальных алгоритмах над дизъюнктивными нормальными формами / Ю. И. Журавлев // Докл. АН СССР. — 1979. — 245:2. — С. 289—292.
60. *Журавлев, Ю. И.* Реализация булевых функций с малым числом нулей дизъюнктивными нормальными формами и смежные задачи / Ю. И. Журавлев, А. Ю. Коган // Докл. АН СССР. — 1985. — 285:4. — С. 795—799.
61. Задачи типа многих коммивояжеров в изменяющихся условиях / М. С. Германчук [и др.] // Сборник материалов международной конференции КРОМШ-2020. — Симферополь : ПОЛИПРИНТ, 2020. — С. 241—245.

62. *Заева, К.* Система поиска минимального пути в среде с полигональными препятствиями / К. Заева, А. Семенов // 24-я Международная конференция по компьютерной графике и зрению: труды конференции. — 2014.
63. *Иванко, Е. Е.* Маршрутно-распределительные задачи: теория и приложения: дис. ... д-та физ.-мат. наук : 01.01.09 / Е. Е. Иванко. — Екатеринбург, 2015. — 289 с.
64. Исследование эвристических алгоритмов в задачах прокладки и оптимизация маршрутов в среде с препятствиями / Р. Нейдорф [и др.]. — 2016. — URL: https://cyberleninka.ru/article/n/issledovanie-evristicheskih-algoritmov-v-zadachah-prokladki-i-optimizatsiya-marshrutov-v-srede-s-prepyatstviyami (дата обр. 01.03.2021).
65. *Карманов, В. Г.* Математическое программирование. Учебное пособие / В. Г. Карманов. — 5 изд., стереотип. — М. : ФИЗМАТЛИТ, 2004. — 264 с.
66. *Касьянов, В. Н.* Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. — СПб. : БХВ-Петербург, 2003. — 1104 с.
67. *Киселев, Д. А.* Игровой подход в задаче двух коммивояжеров / Д. А. Киселев, Т. И. Орлова, М. С. Германчук // Математика, информатика, компьютерные науки, моделирование, образование: сборник научных трудов научно-практической конференции МИКМО-2017 и Таврической научной конференции студентов и молодых специалистов по математике и информатике / Под ред. В. А. Лукьяненко. — Симферополь : ИП Корниенко А. А., 2017. — С. 131—137.
68. *Князь, Д. В.* Методы кластеризации многомерных статистических данных // Молодежь и наука: сборник материалов X Юбилейной Всероссийской научно-технической конференции студентов, аспирантов и молодых ученых с международным участием, посвященной 80-летию образования Красноярского края / Отв. ред. О. А. Краев — Красноярск: Сиб. федер. ун-т / Д. В. Князь. — 2014. — URL: http://elib.sfu-kras.ru/handle/2311/17235 (дата обр. 01.03.2021).
69. *Козлов, И. В.* Устойчивость в задачах поиска минимального разреза на графе / И. В. Козлов // МАИС. — 2014. — Т. 21, № 4. — С. 54—63.
70. *Козлова, М. Г.* Знаниеориентированные модели принятия решений / М. Г. Козлова // Ученые записки СГУ. — 1998. — № 7. — С. 76—83.
71. *Козлова, М. Г.* Многокритериальные модели принятия решений с линейными псевдобулевыми функциями и дизъюнктивным ограничением / М. Г. Козлова // Искусственный интеллект. — 2000. — № 2. — С. 67—73.
72. *Козлова, М. Г.* Синтез сужающих запросов / М. Г. Козлова // Динамические системы. — 2000. — № 16. — С. 208—211.
73. *Козлова, М. Г.* Обобщения задачи коммивояжера: знаниеориентированный подход / М. Г. Козлова, М. С. Германчук // Інформатика та системні науки (ІСН-2013): матеріали IV Всеукр. наук.-практ. конф., (м. Полтава, 21-23 берез. 2013 р.) / за ред. Ємця О. О. — Полтава : ПУЕТ, 2013. — С. 147—150.
74. *Козлова, М. Г.* Прикладные алгоритмы интеллектуализации обработки информации в моделировании задач типа коммивояжера / М. Г. Козлова, М. С. Германчук // Анализ, моделирование, управление, развитие социально-экономических систем: сборник науч-

ных трудов IX Международной школы-симпозиума АМУР-2015, Севастополь, 12-21 сентября 2015 / Под ред. доцента А. В. Сигала. — Симферополь : КФУ имени В. И. Вернадского, 2015. — С. 161—164.

75. *Козлова, М. Г.* Приближенное решение задачи о максимальном разрезе и ее применение / М. Г. Козлова, М. С. Германчук, Э. Д. Куртнебиев // Інформатика та системні науки (ІСН-2013): матеріали IV Всеукр. наук.-практ. конф., (м. Полтава, 21-23 берез. 2013 р.) / за ред. Ємця О. О. — Полтава : ПУЕТ, 2013. — С. 150—153.
76. *Козлова, М. Г.* Система интеллектуализированного анализа и мониторинга распространения и влияния политических интернет-мемов / М. Г. Козлова, В. А. Лукьяненко, О. О. Макаров // Дистанционные образовантельные технологии / Сборник трудов VI Международной научно-практической конференции. — Симферополь : Изд-во ООО «Издательство Типография «Ариал», 2021. — С. 244—251.
77. *Коломейченко, А. А.* Алгоритмы выделения сообществ в социальных сетях / А. А. Коломейченко, А. А. Чеповский, А. М. Чеповский // Фундамент. и прикл. матем. — 2014. — Т. 19, № 1. — С. 21—32.
78. *Корте, Б.* Комбинаторная оптимизация. Теория и алгоритмы. / Пер. с англ. М. А. Бабенко / Б. Корте, Й. Фиген. — М. : МЦНМО, 2015. — 720 с.
79. *Кочетов, Ю. А.* Генетический локальный поиск для задачи о разбиении графа на доли ограниченной мощности / Ю. А. Кочетов, А. В. Плясунов // Журнал выч. математики и матем. физики. — 2012. — Т. 52, № 1. — С. 164—176.
80. *Красовский, Н. Н.* Некоторые задачи теории устойчивости движения / Н. Н. Красовский. — М. : Физматлит, 1959. — 212 с.
81. *Куприянова, Н. И.* Концептуальная модель кластеризации данных / Н. И. Куприянова // Известия ЮФУ. Технические науки. — 2012. — № 4. — С. 256—260.
82. *Курейчик, В. М.* Обзор по применению методов оптимизации пчелинных колоний / В. М. Курейчик, А. А. Кажаров // Международный журнал по информатике и технике. — 2011. — Т. 8, № 3. — С. 3037—3045.
83. *Левитин, А. В.* Алгоритмы: введение в разработку и анализ / А. В. Левитин. — М. : Вильямс, 2006. — 576 с.
84. *Леденева, Т. М.* Синтез нечетких продукционных правил на основе кластеризации наблюдаемых данных / Т. М. Леденева, А. В. Алтухов // Вестник Воронежского государственного технического университета. — 2010. — Т. 6, № 6. — С. 113—117.
85. *Леденева, Т. М.* Эволюционный алгоритм для решения задачи нечеткой кластеризации / Т. М. Леденева, С. А. Моисеев // Вестник Воронежского государственного технического университета. — 2012. — Т. 8, № 2. — С. 4—8.
86. *Леденева, Т. М.* Организация структуры нечеткой базы правил / Т. М. Леденева, М. А. Сергиенко // Информационные технологии. — 2010. — Т. 166, № 6. — С. 46—49.
87. Лекции по теории графов / В. А. Емеличев [и др.]. — Москва : Книжный дом «Либроком», 2009. — 392 с.
88. *Леонтьев, В. К.* Устойчивость в линейных дискретных задачах / В. К. Леонтьев // Проблемы кибернетики. — 1979. — № 35. — С. 169—185.

89. *Леонтьев, В. К.* Устойчивость задачи коммивояжера / В. К. Леонтьев // Ж. вычисл. матем. и матем. физ. — 1975. — Т. 15, № 5. — С. 1298—1309.
90. *Лукъяненко, В. А.* Задачи маршрутизации на сетевых структурах символического образа динамических систем / В. А. Лукьяненко, М. С. Германчук, О. С. Кортнева // «Информационные системы и технологии в моделировании и управлении»: сборник трудов V Международной научно-практической конференции (20-22 мая 2020 г.) / отв. редактор К. А. Маковейчук. — Симферополь : ООО «Издательство типография “Ариал”», 2020. — С. 178—185.
91. *Макаров, О. О.* Разработка алгоритмов маршрутизации в сложных сетях / О. О. Макаров, М. С. Германчук // Математика, информатика, компьютерные науки, моделирование, образование: сборник научных трудов научно-практической конференции МИКМО-2018 и Таврической научной конференции студентов и молодых специалистов по математике и информатике / Под ред. В. А. Лукьяненко. — Симферополь : ИП Корниенко А. А., 2018, Вып. 2. — С. 127—135.
92. *Масич, И. С.* Поисковые алгоритмы условной оптимизации: монография / И. С. Масич. — Красноярск : СибГАУ, 2013. — 160 с.
93. Математика в современном мире. Международная конференция, посвященная 60-летию Института математики им. С. Л. Соболева: тез. докладов / под ред. Г. В. Демиденко. — Новосибирск : Изд-во Института математики, 2017. — 592 с.
94. *Меламед, И. И.* Задача коммивояжера. Вопросы теории / И. И. Меламед, С. И. Сергеев, И. Х. Сигал // Автомат. и телемех. — 1989. — № 9. — С. 3—33.
95. *Меламед, И. И.* Задача коммивояжера. Приближенные алгоритмы / И. И. Меламед, С. И. Сергеев, И. Х. Сигал // Автомат. и телемех. — 1989. — № 11. — С. 3—26.
96. *Меламед, И. И.* Задача коммивояжера. Точные алгоритмы / И. И. Меламед, С. И. Сергеев, И. Х. Сигал // Автомат. и телемех. — 1989. — № 10. — С. 3—29.
97. Методы маршрутной оптимизации радиационно опасных работ / О. Л. Ташлыков [и др.] // Седьмая научно-техническая конференция «Безопасность, эффективность и экономика атомной энергетики». — 2010. — С. 153—156.
98. *Михеенкова, М. А.* Об одном подходе к распознаванию рациональности в коллективах агентов / М. А. Михеенкова, В. К. Финн // Искусственный интеллект и принятие решений. — 2010. — № 3. — С. 22—32.
99. Морфологический анализатор pymorphy2. — 2016. — URL: https : / / pymorphy2 . readthedocs.io/en/latest/ (дата обр. 01.03.2021).
100. *Муравник, А. Б.* Функции Ляпунова в задаче нейросетевого моделирования: сравнительный анализ / А. Б. Муравник // Радиолокация, навигация, связь: сб. трудов XXVI Международной научно-технической конференции (г. Воронеж 29 сентября – 1 октября 2020 г., в 6 т. / Воронежский государственный университет; АО «Созвездие». — Воронеж : Изд. Дом ВГУ, 2020. — С. 49—55.
101. *Панишев, А. В.* Модели и методы оптимизации в задаче коммивояжера / А. В. Панишев, Д. Д. Плечистый. — Житомир : ЖГТУ, 2006. — 300 с.

102. *Пападимитриу, Х.* Комбинаторная оптимизация. Алгоритмы и сложность / Х. Пападимитриу, К. Стайглиц. — М. : Мир, 1984. — 510 с.

103. *Петрунин, С. В.* Использование метода последовательной сепарации для решения задачи коммивояжера / С. В. Петрунин // Научный вестник МГТУ ГА. Серия Менеджмент, Экономика, Финансы. — 2009. — № 146. — С. 87—90.

104. *Поляков, И. В.* Алгоритмы поиска путей на графах большого размера / И. В. Поляков, А. А. Чеповский, А. М. Чеповский // Фундамент. и прикл. матем. — 2014. — Т. 19, № 1. — С. 165—172.

105. *Поспелов, Д. А.* От коллектива автоматов к мультиагентным системам / Д. А. Поспелов // Программные продукты и системы. — 2003. — № 2. — С. 39—44.

106. *Рассел, С.* Искусственный интеллект: современный подход / С. Рассел, П. Норвиг. — 2-е изд. : Пер. с англ. — М. : Издательский дом «Вильямс», 2006. — 1408 с.

107. *Ревотюк, М. П.* Реоптимизация решения задач о назначении / М. П. Ревотюк, П. М. Батура, А. М. Полоневич // Доклады БГУИР. — 2011. — № 1. — С. 55—62.

108. *Ревотюк, М. П.* Реоптимизация решения задач о назначении / М. П. Ревотюк, М. К. Кароли, П. М. Батура // Доклады БГУИР. — 2013. — № 7. — С. 25—31.

109. *Сапоженко, А. А.* О поиске максимального верхнего нуля монотонных функций на ранжированных множествах / А. А. Сапоженко // Ж. вычисл. матем. и матем. физ. — 1991. — Т. 31, № 12. — С. 1871—1884.

110. *Сваами, М.* Графы, сети и алгоритмы / М. Сваами, К. Тхуласираман. — М. : Мир, 1984. — 454 с.

111. *Седжвик, Р.* Алгоритмы на C++ / Пер. с англ. А. А. Моргунова / Р. Седжвик. — М. : ООО «И. Д. Вильямс», 2011. — 1056 с.

112. *Сергеев, С. И.* Гибридные системы управления и динамическая задача коммивояжера / С. И. Сергеев // Автоматика и телемеханика. — 2008. — № 1. — С. 45—54.

113. *Сергеев, С. И.* Математические модели и методы решения задач дискретной оптимизации / С. И. Сергеев. — 2-ое, доп. и перераб. — Киев : Наукова думка, 1988. — 471 с.

114. *Сергиенко, И. В.* Исследование устойчивости и параметрический анализ дискретных оптимизационных задач / И. В. Сергиенко, Л. Н. Козерацкая, Т. Т. Лебедева. — Киев : Наукова думка, 1995. — 170 с.

115. *Сергиенко, И. В.* Задачи целочисленного программирования с неоднозначно заданными данными: точные и приближенные решения / И. В. Сергиенко, Н. В. Семенова // Кибернетика и системный анализ. — 1995. — № 6. — С. 75—86.

116. *Сергиенко, Т. И.* Об устойчивости по ограничениям многокритериальной задачи целочисленного программирования / Т. И. Сергиенко // Доклады АН УССР. — 1989. — Т. А, № 3. — С. 79—81.

117. *Сесекин, А. Н.* Задачи маршрутизации с ограничениями, ориентированные на применение в атомной энергетике / А. Н. Сесекин, А. А. Ченцов, А. Г. Ченцов // Материалы конференции «Дискретная оптимизация и исследование операций». — Новосибирск : Изд-во Ин-та математики, 2010. — С. 154.

118. *Сесекин, А. Н.* Об одной задаче маршрутизации «на узкие места» / А. Н. Сесекин, А. А. Ченцов, А. Г. Ченцов // Тр. ИММ УрО РАН. — 2010. — Т. 16, № 1. — С. 152—170.
119. *Сесекин, А. Н.* Методы маршрутизации и их приложения в задачах повышения безопасности и эффективности эксплуатации атомных станций / А. Н. Сесекин, О. Л. Ташлыков, П. А. Ченцов. — М. : Новые технологии, 2012. — 234 с.
120. *Силин, И.* Обзор и экспериментальное сравнение алгоритмов кластеризации графов / И. Силин, М. Панов // Сборник трудов 39-й междисциплинарной школы-конференции ИППИ РАН «Информационные технологии и системы 2015». — М. : ИППИ РАН, 2015. — С. 1042—1059.
121. Слабое и сильное определение интеллектуального агента. Портал искусственного интеллекта. — URL: http://www.aiportal.ru/articles/multiagent-systems/weak-and-strong-intelligent-agent.html (дата обр. 01.03.2021).
122. *Смирнов, А. В.* Модели формирования коалиций между кооперативными агентами: состояние и перспективы исследований / А. В. Смирнов, Л. Б. Шереметов // Искусственный интеллект и принятие решений. — 2011. — № 1. — С. 36—48.
123. *Стальский, В. В.* Эволюционный метод когнитивного развития обучающихся автоматных агентов / В. В. Стальский // Искусственный интеллект и принятие решений. — 2014. — № 5. — С. 52—62.
124. *Степкин, А. В.* Распознавание конечных графов тремя агентами / А. В. Степкин // Искусственный интеллект. — 2011. — № 2. — С. 84—93.
125. *Стефанюк, В. Л.* Локальная организация интеллектуальных систем / В. Л. Стефанюк. — М. : ФИЗМАТЛИТ, 2004. — 328 с.
126. *Супруненко, Д. А.* К задаче о бродячем торговце / Д. А. Супруненко // Кибернетика и системный анализ. — 1975. — № 5. — С. 121—128.
127. *Тарасов, В. Б.* Искусственная жизнь и нечеткие эволюционные многоагентные системы — основные теоретические подходы к построению интеллектуальных организаций / В. Б. Тарасов // Известия РАН: Теория и системы управления. — 1998. — № 5. — С. 12—23.
128. *Тимофеева, Н. К.* Метод структурно-алфавитного поиска и подклассы разрешимых задач из класса задачи коммивояжера / Н. К. Тимофеева // УСиМ. — 2008. — № 4. — С. 20—36.
129. *Турбуланов, П. А.* Эволюционные алгоритмы решения задачи многих коммивояжеров / П. А. Турбуланов, М. С. Германчук // Математика, информатика, компьютерные науки, моделирование, образование: сборник научных трудов научно-практической конференции МИКМО-2018 и Таврической научной конференции студентов и молодых специалистов по математике и информатике / Под ред. В. А. Лукьяненко. — Симферополь : ИП Корниенко А. А., 2018, Вып. 1. — С. 163—168.
130. *Узарашвили, В. М.* Взаимодействие интеллектуальных агентов в решении сетевых задач. Обзор подходов / В. М. Узарашвили, М. С. Германчук // Математика, информатика, компьютерные науки, моделирование, образование: сб. науч. трудов научно-практической конференции МИКМО-2017 и Таврической научной конферен-

ции студентов и молодых специалистов по математике и информатике / Под ред. В. А. Лукьяненко. — Симферополь : ИП Корниенко А. А., 2017. — С. 172—177.

131. *Ураков, А. Р.* Использование особенностей взвешенных графов для более быстрого определения их характеристик / А. Р. Ураков, Т. В. Тимеряев // ПДМ. — 2012. — № 2. — С. 95—99.

132. *Фирюлина, О. С.* Алгоритмы поиска максимальных независимых множеств графа и экспериментальная оценка их эффективности: дис. ... к-та физ.-мат. наук : 05.13.18 / О. С. Фирюлина. — Санкт-Петербург, 2014. — 149 с.

133. *Черняхівський, В.* Серединні умови побудови максимальних простих ланцюгів неповного графа / В. Черняхівський // Вісн. Львів. ун-ту. Сер. прикл. математики та інформатики. — 2008. — № 14. — С. 204—209.

134. *Чесноков, В. О.* Выделение сообществ в социальных графах по множеству признаков с частичной информацией / В. О. Чесноков, П. Г. Ключарев // Машиностроение и компьютерные технологии. — 2015. — № 9. — С. 188—199.

135. *Шило, В. П.* Метод глобального равновесного поиска решения задачи о максимальном взвешенном разрезе графа / В. П. Шило, О. В. Шило, В. А. Рощин // Кибернетика и системный анализ. — 2012. — Т. 48, № 4. — С. 101—105.

136. *Щербина, О. А.* Метаэвристические алгоритмы для задач комбинаторной оптимизации / О. А. Щербина // Таврические вестник информатики и математики. — 2014. — № 1. — С. 56—72.

137. *Яблонский, С. В.* Введение в дискретную математику : Учеб. пособие для вузов / Под. ред. В. А. Садовничего. – 3-е изд., стер. / С. В. Яблонский. — М : Высш. шк., 2001. — 384 с.

138. A genetic algorithm with new local operators for multiple traveling salesman problems / K.-M. Lo [et al.] // International Journal of Computational Intelligence Systems. — 2018. — Vol. 11. — P. 692—705.

139. A minimal technology routing system for meals on wheels / J. Bartholdi [et al.] // Interfaces. — 1983. — Vol. 13, no. 3. — P. 1—8.

140. A multiple traveling salesman problem model for hot rolling scheduling in Shanghai Baoshan Iron & Steel Complex / T. Lixin [et al.] // Eur. J. Oper. Res. — 2000. — Vol. 124, no. 2. — P. 267—282.

141. A novel hybrid approach for solving the multiple traveling salesmen problem / Y. Harrath [et al.] // Arab Journal of Basic and Applied Sciences. — 2019. — Jan. — Vol. 26. — P. 103—112.

142. A review of bio-inspired algorithms as image processing techniques / N. E. Abdul Khalid [et al.] // Communications in Computer and Information Science. — 2011. — Vol. 179. — P. 660—673.

143. A theoretical framework of hybrid approaches to MAX SAT / T. Asano [et al.] // Algorithms and Computation / ed. by H. W. Leong, H. Imai, S. Jain. — Berlin, Heidelberg : Springer Berlin Heidelberg, 1997. — P. 153—162.

144. *Aditi, K.* A hybrid heuristic algorithm for single and multi-objective imprecise traveling salesman problems / K. Aditi, M. K. Manas, M. Manoranjan // Journal of Intelligent and Fuzzy Systems. — 2016. — Jan. — Vol. 30, no. 4. — P. 1987—2001.

145. *Ahn, Y.-Y.* Link communities revel multiscale complexity in networks / Y.-Y. Ahn, J. Bagrow, S. Lehmann // Nature. — 2010. — Vol. 466. — P. 761—764.

146. *Ali, A. I.* The asymmetric m-traveling salesmen problem: a duality based branch-and-bound algorithm / A. I. Ali, J. L. Kennington // Discrete Applied Mathematics. — 1986. — Vol. 13, no. 2. — P. 259—276.

147. An algorithm for the traveling salesman problem / J. D. C. Little [et al.] // Operations Research. — 1963. — No. 11. — P. 972—989.

148. An effective parallel approach to solve multiple traveling salesmen problem / A. Othman [et al.] // Advanced Intelligent Systems for Sustainable Development (AI2SD'2018). — Ezziyyani, Mostafa. — Cham : Springer International Publishing, 2019. — P. 647—664.

149. An efficient method for segmentation of images based on fractional calculus and natural selection / P. Ghamisi [et al.] // Expert Syst. Appl. — USA, 2012. — Vol. 39, no. 16. — P. 12407—12417.

150. Approximation algorithms for connected graph factors of minimum weight / K. Cornelissen [et al.] // Theory of Computing Systems. — 2018. — Vol. 62. — P. 441—464.

151. *Archetti, C.* Reoptimizing the traveling salesman problem / C. Archetti, L. Bertazzi, M. G. Speranza // Networks. — 2003. — Oct. — Vol. 42. — P. 154—159.

152. *Ausiello, G.* On-Line Algorithms / G. Ausiello, L. Becchetti // Paradigms of combinatorial optimization. — John Wiley, Sons, Ltd, 2013. — Chap. 15. P. 473—509.

153. *Bartholdi, J. J.* An O(NlogN) planar travelling salesman heuristic based on spacelling curves / J. J. Bartholdi, L. K. Platzman // Operations Research Letters. — 1982. — Vol. 1, no. 4. — P. 121—125.

154. *Basu, S.* Tabu search implementation on traveling salesman problem and its variations: a literature survey / S. Basu // American Journal of Operations Research. — 2012. — Jan. — Vol. 02.

155. *Bektas, T.* The multiple traveling salesman problem: an overview of formulations and solution procedures / T. Bektas // Omega. — 2006. — June. — Vol. 34. — P. 209—219.

156. *Bellmore, M.* Transformation of multisalesman problem to the standard traveling salesman problem / M. Bellmore, S. Hong // J. ACM. — 1974. — Vol. 21. — P. 500—504.

157. *Berg, T.* Reoptimization of traveling salesperson problems: changing single edge-weights / T. Berg, H. Hempel // Language and Automata Theory and Applications / ed. by A. H. Dediu, A. M. Ionescu, C. Martín-Vide. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2009. — P. 141—151.

158. *Bertsimas, D.* On the spacefilling curve heuristic for the Eculidean traveling salesman problem / D. Bertsimas, M. Grigni // Operations Research Letters. — 1989. — Vol. 8, no. 5. — P. 241—244.

159. *Bertsimas, D.* On the spacefilling curve heuristic for the Euclidean traveling salesman problem / D. Bertsimas, P. Jaillet, A. Odoni // Operations Research Letters. — 1990. — Vol. 38, no. 6. — P. 1019—1033.

160. *Blei, D.* Latent Dirichlet allocation / D. Blei, A. Ng, M. I. Jordan // J. Mach. Learn. Res. — 2003. — Vol. 3. — P. 993—1022.

161. *Blum, C.* Ant colony optimization: introduction and recent trends / C. Blum // Physics of Life Reviews. — 2005. — Dec. — Vol. 2. — P. 353—373.

162. *Böckenhauer, H.-J.* Reoptimization of the metric deadline TSP / H.-J. Böckenhauer, D. Komm // Mathematical Foundations of Computer Science 2008 / ed. by E. Ochmański, J. Tyszkiewicz. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. — P. 156—167.

163. *Boros, E.* Pseudo-Boolean optimization / E. Boros, P. Hammer // Discret. Appl. Math. — 2002. — Vol. 123. — P. 155—225.

164. *Bubeck, S.* Introduction to online optimization / S. Bubeck. — 2011. — URL: http://sbubeck.com/BubeckLectureNotes.pdf (visited on 03/01/2021).

165. *Buchbinder, N.* The design of competitive online algorithms via a primal–dual approach / N. Buchbinder, J. Naor // Foundations and Trends® in Theoretical Computer Science. — 2009. — Vol. 3, no. 2/3. — P. 93—263.

166. *Carter, A. E.* A new approach to solving the multiple traveling salesperson problem using genetic algorithms / A. E. Carter, C. T. Ragsdale // European Journal of Operational Research. — 2006. — Vol. 175, no. 1. — P. 246—257.

167. *Chin, F.* Algorithms for updating minimal spanning trees / F. Chin, D. Houck // J. Comput. Syst. Sci. — 1978. — Vol. 16. — P. 333—344.

168. *Crama, Y.* Boolean functions: theory, algorithms, and applications / Y. Crama, P. Hammer. — 01/2011.

169. *Croes, G. A.* A method for solving traveling-salesman problems / G. A. Croes // Operations Research. — 1958. — Vol. 6. — P. 791—812.

170. *Deineko, V.* Sometimes travelling is easy: the master tour problem / V. Deineko, R. Rudolf, G. Woeginger // SIAM J. Discret. Math. — 1998. — Vol. 11. — P. 81—93.

171. *Deineko, V.* Fast minimum-weight double-tree shortcutting for metric TSP / V. Deineko, A. Tiskin // WEA. — 2007.

172. *Demetrescu, C.* A new approach to dynamic all pairs shortest paths / C. Demetrescu, G. Italiano // J. ACM. — 2004. — Jan. — Vol. 51. — P. 968—992.

173. *Donskoy, V. I.* A synthesis of pseudo-Boolean empirical models by precedential information / V. I. Donskoy // Vestnik YuUrGU. Ser. Mat. Model. Progr. — 2018. — Vol. 11, no. 2. — P. 96—107.

174. *Donskoy, V.* Multiple criteria models with the linear pseudo-Boolean functions and disjunctive restrictions / V. Donskoy, I. Perekhod // Multiple Criteria Decision Making / ed. by G. Fandel, T. Gal. — Berlin, Heidelberg : Springer Berlin Heidelberg, 1997. — P. 13—21.

175. *Dorigo, M.* Ant colony optimization / M. Dorigo, T. Stutzle. — Cambridge : Bradford Book, 2004.

176. *Ebenegger, C.* Pseudo-Boolean functions and stability of graphs / C. Ebenegger, P. Hammer, D. Werra // North-holland Mathematics Studies. — 1984. — Vol. 95. — P. 83—97.

177. *Escoffier, B.* Complexity and approximation in reoptimization / B. Escoffier, V. Bonifaci, G. Ausiello. — 2011. — Feb.

178. *Feng, Y.* Ant colony optimization for image segmentation / Y. Feng, Z. Wang // Ant Colony Optimization / ed. by A. Ostfeld. — Rijeka : IntechOpen, 2011. — Chap. 17.

179. *Fiechter, C.-N.* A parallel tabu search algorithm for large traveling salesman problems / C.-N. Fiechter // Discrete Applied Mathematics. — 1994. — Vol. 51, no. 3. — P. 243—267.

180. *Foldes, S.* Disjunctive and conjunctive normal forms of pseudo-Boolean functions / S. Foldes, P. Hammer // Discret. Appl. Math. — 2000. — Vol. 107. — P. 1—26.

181. *Fowlkes, C.* Efficient spatiotemporal grouping using the Nystrom method / C. Fowlkes, S. Belongie, J. Malik // Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. Vol. 1. — 2001. — P. I—I.

182. *Froushani, M.* Development of an innovative algorithm for the traveling salesman problem (TSP) / M. Froushani, R. Yusuff // European Journal of Scientific Research. — 2009. — Apr. — Vol. 29.

183. *Gavish, B.* An optimal solution method for Large–Scale multiple traveling salesmen problems / B. Gavish, K. Srikanth // Operations Research. — 1986. — Sept. — Vol. 34. — P. 698—717.

184. Genetic algorithm for solving multiple traveling salesmen problem using a new crossover and population generation / D. Singh [et al.] // Computación y Sistemas. — 2018. — Vol. 22, no. 2. — P. 491—503.

185. *Germanchuk, M. S.* Identification and prediction of an internet meme flow life cycle / M. S. Germanchuk, M. G. Kozlova, V. A. Lukianenko // CEUR Workshop Proceedings. — 2021. — Vol. 2914. — P. 112—123.

186. *Germanchuk, M. S.* Some features of design of intelligent systems for processing the internet memes flow / M. S. Germanchuk, M. G. Kozlova, V. A. Lukianenko // CEUR Workshop Proceedings. — 2021. — Vol. 2834. — P. 148—158.

187. *Gilmore, P.* Sequencing a one state-variable machine: a solvable case of the traveling salesman problem / P. Gilmore, R. Gomory // Operations Research. — 1964. — Oct. — Vol. 12, no. 5. — P. 655—679.

188. *Goldberg, D.* Alleles, Loci and the traveling salesman problem / D. Goldberg, R. Lingle // Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications. — Los Angeles, 1985. — P. 154—159.

189. *Gromicho, J.* Exact solution of multiple traveling salesman problems / J. Gromicho, J. Paixão, I. Bronco // Combinatorial Optimization / ed. by M. Akgül, H. W. Hamacher, S. Tüfekçi. — Berlin, Heidelberg : Springer Berlin Heidelberg, 1992. — P. 291—292.

190. *Hammer, P.* Boolean methods in operations research and related areas / P. Hammer, S. Rudeanu. — 1st ed. — Springer–Verlag Berlin Heidelberg, 1968.

191. *Hammer, P.* Pseudo-Boolean methods for bivalent programming : lecture at the first European meeting of the Institute of Management Sciences and of the Econometric Institute, Warsaw, September 2–7, 1966 / P. Hammer, S. Rudeanu.

192. *Hammer, P. L.* Pseudo-Boolean remarks on balanced graphs / P. L. Hammer // Numerische Methoden bei Optimierungsaufgaben Band 3: Optimierung bei graphentheoretischen und ganzzahligen Problemen / ed. by L. Collatz, G. Meinardus, W. Wetterling. — Basel : Birkhäuser Basel, 1977. — P. 69—78.

193. *Hatamlou, A.* A combined approach for clustering based on K-means and gravitational search algorithms / A. Hatamlou, S. Abdullah, H. Nezamabadi-pour // Swarm and Evolutionary Computation. — 2012. — Vol. 6. — P. 47—52.

194. *Helsgaun, K.* An effective implementation of the Lin–Kernighan traveling salesman heuristic / K. Helsgaun // Eur. J. Oper. Res. — 2000. — Vol. 126. — P. 106—130.

195. *Helsgaun, K.* An extension of the Lin–Kernighan–Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems: technical report / K. Helsgaun. — 2017.

196. *Huizing, D.* Solving the mTSP for fresh food delivery / D. Huizing. — 2015. — URL: https://repository.tudelft.nl/islandora/object/uuid%3A8af405cc-bdd1-46c0-a790-a66471eadb3f (visited on 03/01/2021).

197. Hybridizing genetic algorithms with branch and bound techniques for the resolution of the TSP / C. Cotta [et al.] // Artificial Neural Nets and Genetic Algorithms. — Vienna : Springer, 1995. — P. 277—280.

198. Improved lower bounds for the universal and a priori TSP / I. Gorodezky [et al.] // Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques / ed. by M. Serna [et al.]. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. — P. 178—191.

199. *Jaillet, P.* Online vehicle routing problems: a survey / P. Jaillet, M. R. Wagner // The Vehicle Routing Problem: Latest Advances and New Challenges / ed. by B. Golden, S. Raghavan, E. Wasil. — Boston, MA : Springer US, 2008. — P. 221—237.

200. *Jones, K.* A statistical interpretation of term specificity and its application in retrieval / K. Jones // J. Documentation. — 2004. — Vol. 60. — P. 493—502.

201. *Junjie, P.* An ant colony optimization algorithm for multiple travelling salesman problem / P. Junjie, W. Dingwei // First International Conference on Innovative Computing, Information and Control — Volume I (ICICIC'06). — 2006. — P. 210—213.

202. *Kalmanson, K.* Edgeconvex circuits and the traveling salesman problem / K. Kalmanson // Canadian Journal of Mathematics. — 1975. — Vol. 27, no. 5. — P. 1000—1010.

203. *Kara, I.* Integer linear programming formulations of multiple salesman problems and its variations / I. Kara, T. Bektas // Eur. J. Oper. Res. — 2006. — Vol. 174. — P. 1449—1458.

204. *Karaboga, D.* A survey: algorithms simulating bee swarm intelligence / D. Karaboga, B. Akay // Artificial Intelligence Review. — 2009. — Vol. 31. — P. 61—85.

205. *Keuchel, J.* Efficient graph cuts for unsupervised image segmentation using probabilistic sampling and SVD-based approximation / J. Keuchel, C. Schnorr // Third International Workshop on statistical and computational theories of Vision. — 2003. — P. 120—128.

206. *King, V.* Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs / V. King // 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039). — 1999. — P. 81—89.

207. *Király, A.* Optimization of multiple traveling salesmen problem by a novel representation based genetic algorithm / A. Király, J. Abonyi // Intelligent Computational Optimization in Engineering: Techniques and Applications / ed. by M. Köppen, G. Schaefer, A. Abraham. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. — P. 241—269.

208. *Kirkpatrick, S.* Optimization by simulated annealing / S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi // Science. — 1983. — Vol. 220, no. 4598. — P. 671—680.

209. *Korte, B.* Combinatorial optimization: theory and algorithms / B. Korte, J. Vygen. — Springer, 2007.

210. *Kozlova, M. G.* Building a transport network model using satellite images / M. G. Kozlova, M. S. Germanchuk // Taurida Journal of Computer Science Theory and Mathematics. — 2020. — Vol. 47, no. 2. — P. 7—18.

211. *Kozlova, M. G.* Development of the toolkit to process the internet memes meant for the modelling, analysis, monitoring and management of social processes / M. G. Kozlova, V. A. Lukianenko, M. S. Germanchuk // "Recognition and Perception of Images. Fundamentals and Applications" / ed. by I. B. Abbasov. — USA : Wiley, 2021. — P. 189—220.

212. *Krishna, H.* An approach for solving multiple travelling salesman problem using ant colony optimization / H. Krishna, R. Singh // Computer Engineering and Intelligent Systems. — 2015. — Vol. 6. — P. 13—17.

213. *Land, A. H.* An automatic method of solving discrete programming problems / A. H. Land, A. G. Doig // Econometrica. — 1960. — July. — Vol. 28, no. 3. — P. 497—520.

214. *Laporte, G.* Cutting planes algorithm for the m-salesmen problem / G. Laporte, Y. A. Nobert // J. Oper. Res. Soc. — 1980. — Vol. 31, no. 11. — P. 1017—1023.

215. *Laptik, R.* Application of ant colony optimization for image segmentation / R. Laptik, D. Navakauskas // Elektronika ir Elektrotechnika. — 2007. — Vol. 80, no. 8. — P. 13—18.

216. *Libura, M.* Sensitivity analysis for minimum Hamiltonian path and traveling salesman problems / M. Libura // Discret. Appl. Math. — 1991. — Vol. 30. — P. 197—211.

217. *Libura, M.* Optimality conditions and sensitivity analysis for combinatorial optimization problems / M. Libura // Control and Cybernetics. — 1996. — Jan. — Vol. 25.

218. *Lin, S.* An effective heuristic algorithm for the traveling-salesman problem / S. Lin, B. Kernighan // Oper. Res. — 1973. — Vol. 21. — P. 498—516.

219. *Loria, S.* Tutorial: finding important words in text using TF-IDF / S. Loria. — URL: https://stevenloria.com/tf-idf (visited on 03/01/2021).

220. *Majid, Y. K.* Modification of the ant colony optimization for solving the multiple traveling salesman problem / Y. K. Majid, D. Farzad, R. Farhad // Romanian Journal of Information Science and Technology. — 2013. — Vol. 16, no. 1. — P. 65—80.

221. *Miller, D. L.* Exact solution of large asymmetric traveling salesman problems / D. L. Miller, J. F. Pekny // Science. — 1991. — Vol. 251, no. 4995. — P. 754—761.

222. Multilevel Image segmentation based on fractional-order Darwinian particle swarm optimization / P. Ghamisi [et al.] // IEEE Transactions on Geoscience and Remote Sensing. — 2014. — Vol. 52, no. 5. — P. 2382—2394.

223. National Traveling Salesman Problems. — URL: http://www.math.uwaterloo.ca/tsp/world/countries.html (visited on 03/01/2021).

224. *Necula, R.* Balancing the subtours for multiple TSP approached with ACS: clustering-based approaches Vs. MinMax formulation / R. Necula, M. Raschip, M. Breaban. — 2018. — Jan.

225. *Oliver, I. M.* A sudy of permutation cossover operators on the traveling salesman problem / I. M. Oliver, D. J. Smith, J. R. Holland // ICGA. — 1987. — P. 224—230.

226. *Platzman, L. K.* Spacefilling curves and the planar travelling salesman problem. Vol. 36 / L. K. Platzman, J. J. Bartholdi. — New York, NY, USA : Association for Computing Machinery, 10/1989. — P. 719—737.

227. Proof verification and the hardness of approximation problems / S. Arora [et al.] // Journal of the ACM (JACM). — 2001. — Sept. — Vol. 45. — P. 501—555.

228. *Ramos, V.* Image colour segmentation by genetic algorithms / V. Ramos, F. Muge. — 2005.

229. *Reinelt, G.* TSPLIB—A traveling salesman problem library / G. Reinelt // ORSA Journal on Computing. — 1991. — Vol. 3, no. 4. — P. 376—384.

230. Reoptimization of minimum and maximum traveling salesman's tours / G. Ausiello [et al.] // Algorithm Theory – SWAT 2006 / ed. by L. Arge, R. Freivalds. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. — P. 196—207.

231. Reusing optimal TSP solutions for locally modified input instances / H.-J. Böckenhauer [et al.] // Fourth IFIP International Conference on Theoretical Computer Science- TCS 2006 / ed. by G. Navarro, L. Bertossi, Y. Kohayakawa. — Boston, MA : Springer US, 2006. — P. 251—270.

232. *Saatchi, S.* Swarm intelligence and image segmentation / S. Saatchi, C.-C. Hung // Swarm Intelligence / ed. by F. T. S. Chan, M. K. Tiwari. — Rijeka : IntechOpen, 2007. — Chap. 10.

233. *Schäffter, M. W.* Scheduling with forbidden sets / M. W. Schäffter // Discrete Applied Mathematics. — 1997. — Vol. 72, no. 1. — P. 155—166. — Models and Algorithms for Planning and Scheduling Problems.

234. *Schalekamp, F.* Algorithms for the universal and a priori TSP / F. Schalekamp, D. Shmoys // Oper. Res. Lett. — 2008. — Jan. — Vol. 36. — P. 1—3.

235. *Schrijver, A.* Combinatorial Optimization. A / A. Schrijver. — Berlin : Springer, 2003.

236. *Sedighpour, M.* An effective genetic algorithm for solving the multiple traveling salesman problem / M. Sedighpour, M. Yousefikhoshbakht, M. D. Narges // Journal of Optimization in Industrial Engineering. — 2012. — Vol. Volume 4, no. 8. — P. 73—79.

237. *Shalev-Shwartz, S.* Online learning and online convex optimization / S. Shalev-Shwartz. — 2012.

238. *Sharon, E.* Fast multiscale image segmentation / E. Sharon, A. Brandt, R. Basri // Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662). Vol. 1. — 2000. — 70—77 vol.1.

239. *Shi, J.* Normalized cuts and image segmentation / J. Shi, J. Malik // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2000. — Vol. 22, no. 8. — P. 888—905.

240. *Shmoys, D.* A constant approximation algorithm for the a priori traveling salesman problem / D. Shmoys, K. Talwar // Integer Programming and Combinatorial Optimization / ed. by A. Lodi, A. Panconesi, G. Rinaldi. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. — P. 331—343.

241. *Sho, S.* Ant colony optimization using genetic information for TSP / S. Sho, M. Haruna, N. Yoshifumi // Proceedings of the International Symposium on Nonlinear Theory and its Applications OLTA2011. — Japan : Kobe. — P. 48—51.

242. Solving multiple traveling salesman problem using the gravitational emulation local search algorithm / R. A. Shokouhi [et al.] // Applied Mathematics. — 2015. — Vol. 9, no. 2. — P. 699—709.

243. *Spira, P. M.* On finding and updating shortest paths and spanning trees / P. M. Spira, P. Pan // 14th Annual Symposium on Switching and Automata Theory (swat 1973). — 1973. — P. 82—84.

244. Stability aspects of the traveling salesman problem based on k-best solutions / M. Libura [et al.] // Discrete Applied Mathematics. — 1998. — Vol. 87, no. 1. — P. 159—185.

245. *Supnick, F.* Extreme hamiltonian lines / F. Supnick // Annals of Mathematics. — 1957. — July. — Vol. 66, no. 1. — P. 179—201.

246. Texture segmentation by multiscale aggregation of filter responses and shape elements / M. Galun [et al.] // Proceedings Ninth IEEE International Conference on Computer Vision. — 2003. — 716—723 vol.1.

247. TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types. — URL: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ (visited on 03/01/2021).

248. *Van der Poort, E. S.* Aspects of sensitivity analysis for the traveling salesman problem : PhD thesis / Van der Poort E. S. — The Netherlands : Department of Econometrics, Operations Research, University of Groningen, 1997.

249. *Veksler, O.* Image segmentation by nested cuts / O. Veksler // Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662). Vol. 1. — 2000. — 339—344 vol.1.

250. *Venkatesh, P.* Two metaheuristic approaches for the multiple traveling salesperson problem / P. Venkatesh, A. Singh // Applied Soft Computing. — 2015. — Vol. 26. — P. 74—89.

251. Vertebrate pollinators: phase transition in a time-dependent generalized traveling-salesperson problem / M. Jungsbluth [et al.] // arXiv: Biological Physics. — 2018.

252. *Xie, J.* Overlapping community detection in networks: the state of the art and comparative study / J. Xie, S. Kelley, B. K. Szymanski // CoRR. — 2011. — Vol. abs/1110.5813.

253. *Yousefikhoshbakht, M.* A combination of sweep algorithm and elite ant colony optimization for solving the multiple traveling salesman problem / M. Yousefikhoshbakht, M. Sedighpour // Proceedings of the Romanian Academy — Series A: Mathematics, Physics, Technical Sciences, Information Science. — 2012. — Dec. — Vol. 13. — P. 295—301.

254. *Yuan, X.* Automatic segmentation of skin lesion images using evolution strategies / X. Yuan, N. Situ, G. Zouridakis // Biomedical Signal Processing and Control. — 2008. — Vol. 3, no. 3. — P. 220—228.

255. *Zhou, H.* A comparative study of improved GA and PSO in solving multiple traveling salesmen problem / H. Zhou, M. Song, W. Pedrycz // Applied Soft Computing. — 2018. — Vol. 64. — P. 564—580.

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021681822

«Программа выбора наилучших туристических маршрутов по Крыму»

Правообладатели: ***Германчук Мария Сергеевна (RU), Козлова Маргарита Геннадьевна (RU), Лукьяненко Владимир Андреевич (RU)***

Авторы: ***Германчук Мария Сергеевна (RU), Козлова Маргарита Геннадьевна (RU), Лукьяненко Владимир Андреевич (RU)***

Заявка № **2021681336**
Дата поступления **14 декабря 2021 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***27 декабря 2021 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0x02A5CFBC00B1ACF59A40A2F08092E9A118
Владелец **Ивлиев Григорий Петрович**
Действителен с 15.01.2021 по 15.01.2035

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU2021681822

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ

Номер регистрации (свидетельства):
2021681822
Дата регистрации: **27.12.2021**
Номер и дата поступления заявки:
2021681336 14.12.2021
Дата публикации и номер бюллетеня:
27.12.2021 Бюл. № 1
Контактные реквизиты:
Лукьяненко Владимир Андреевич,
+79787291655, art-inf@yandex.ru

Автор(ы):
Германчук Мария Сергеевна (RU),
Козлова Маргарита Геннадьевна (RU),
Лукьяненко Владимир Андреевич (RU)
Правообладатель(и):
Германчук Мария Сергеевна (RU),
Козлова Маргарита Геннадьевна (RU),
Лукьяненко Владимир Андреевич (RU)

Название программы для ЭВМ:
«Программа выбора наилучших туристических маршрутов по Крыму»

Реферат:
Программная реализация веб-приложения по построению оптимальных маршрутов по Крыму на основе сервиса Яндекс.Карты и данных существующих организаций из сервиса Яндекс.Справочник. В работе рассмотрены особенности работы с этими сервисами и принципы построения интерфейса вокруг этих сервисов. Цель программы: на основе данных о достопримечательностях, полученных из сервиса Яндекс.Справочник, а также на основе данных пользователя, составить оптимальный маршрут на несколько дней по достопримечательностям, с учетом их времени работы и желаемом времени посещения. Тип ЭВМ: IBM PC - совмест. ПК. ОС: Windows Vista/7/8/10.

Язык программирования: JavaScript, TypeScript

Объем программы для ЭВМ: 4 775 936 Байт

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2022614174

«Программа многоагентной инфраструктурной маршрутизации»

Правообладатели: ***Германчук Мария Сергеевна (RU), Лукьяненко Владимир Андреевич (RU), Макаров Олег Олегович (RU)***

Авторы: ***Германчук Мария Сергеевна (RU), Лукьяненко Владимир Андреевич (RU), Макаров Олег Олегович (RU)***

Заявка № **2022613404**
Дата поступления **09 марта 2022 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***17 марта 2022 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 68b80077e14e40f0a94edbd24145d5c7
Владелец Зубов Юрий Сергеевич
Действителен с 2.03.2022 по 26.05.2023

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

RU 2022614174

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) **ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
2022614174

Дата регистрации: **17.03.2022**

Номер и дата поступления заявки:
2022613404 09.03.2022

Дата публикации: **17.03.2022**

Контактные реквизиты:
Лукьяненко Владимир Андреевич, тел.: +79787291655, адрес электронной почты: art-inf@yandex.ru

Авторы:
Германчук Мария Сергеевна (RU),
Лукьяненко Владимир Андреевич (RU),
Макаров Олег Олегович (RU)

Правообладатели:
Германчук Мария Сергеевна (RU)
Лукьяненко Владимир Андреевич (RU)
Макаров Олег Олегович (RU)

Название программы для ЭВМ:
«Программа многоагентной инфраструктурной маршрутизации»

Реферат:
Программа предназначена для выбора наилучших маршрутов для многих агентов-коммивояжеров в случае структурных преобразований сложной инфраструктурной сети. Многоагентный подход в сочетании с упрощением структуры и кластеризацией сети позволяют получить предварительный набор маршрутов многих коммивояжеров, реоптимизация которых позволяет строить новые маршруты в случае изменения структуры сети. Программная реализация является решением задачи mTSP для г. Ялты с прилегающими территориями в случае нескольких агентов-коммивояжеров. Многоагентность данных позволяет планировать реализацию функционала взаимодействия в MAC для имитации режимов чрезвычайных ситуаций. Тип ЭВМ: IBM PC-совмест. ПК на базе процессора Intel или AMD; ОС: Windows Vista/7/8/10.

Язык программирования: Python

Объем программы для ЭВМ: 253396 байт

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Крымский федеральный университет имени В. И. Вернадского»
(ФГАОУ ВО «КФУ им. В. И. Вернадского»)

УТВЕРЖДАЮ
Проректор по научной деятельности
д. м. н., проф.
_____ А. В. Кубышкин
«___» _____ 2022 г.

АКТ
о внедрении результатов диссертационного исследования

Результаты диссертационного исследования по теме «Знаниеориентированные модели многоагентной маршрутизации», выполненной Германчук Марией Сергеевной на кафедре информатики Физико-технического института, внедрены в учебный процесс на кафедре политических наук и международных отношений философского факультета Таврической академии в рамках курса «Математические методы в политических исследованиях» (преподаватель М.В. Гаспарян). Результаты диссертационного исследования, в частности, апробированы в процессе работы над грантом РФФИ 21-011-31733/21 «Разработка программного комплекса для автоматического мониторинга влияния политических мемов на русскоязычный сегмент Интернета». А именно Германчук М.С.

- обоснован и программно реализован алгоритм интеллектуализированной обработки информации (изображений) из социальных сетей,
- разработан знаниеориентированный алгоритм анализа семантики новостных заголовков сайтов (в целью формирования запросов),
- предложен алгоритм кластеризации сообществ социальных сетей с целью выявления групп, подверженных влиянию.

Германчук М.С. является одним из авторов разработанного программного комплекса интеллектуальной автоматизированной системы Memometrix, который позволяет мониторить влияние актуальных (политических) мемов на русскоязычный сегмент Интернета. На выходе пользователю предоставляется аналитическая записка с характеристиками наиболее актуальных политических мемов за анализируемый период.

Руководитель проекта,
декан философского факультета
Таврической академии,
доктор философских наук, профессор _____ / О. А. Габриелян /