Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Солдатенко Александр Александрович

РАЗРАБОТКА АЛГОРИТМОВ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ ДЛЯ АНАЛИЗА ГРАФОВЫХ И ГИПЕРГРАФОВЫХ СЕТЕЙ

Специальность 05.13.17 — Теоретические основы информатики

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель
кандидат физико-математических наук,
доцент Семенова Дарья Владиславовна

Красноярск 2021

Содержание

ВВЕДЕНИЕ

Актуальность темы исследования.

В настоящее время графы и гиперграфы используются для представления сетей из различных предметных областей. Графовые модели используются для представления дорожных, мультисервисных и телекоммуникационных сетей [40, 57, 65, 68, 77, 78, 88]. Гиперграфы являются расширением классической теории графов на случай, когда ребро графа может содержать число вершин, отличное от двух [11]. Традиционно гиперграфы нашли практическое применение в комбинаторной химии и разработке реляционных баз данных [18, 69]. Возможность объединения нескольких вершин одним ребром дает сильный инструмент для исследования и анализа процессов в различных сетях. В настоящее время гиперграфы активно применяются при моделировании дорожных и телекоммуникационных сетей, а также для построения семантических сетей при обработке текстов на естественном языке [20, 54, 55, 59, 71, 73, 80, 89, 90]. Актуальным направлением исследований является анализ, проектирование и реструктуризация таких сетей. Значительная часть задач анализа графовых и гиперграфовых сетей сводится к проблемам определения различных конфигураций [35]. Под конфигурацией понимается любая система подмножеств конечного множества. Особо интересными конфигурациями являются кратчайшие пути и максимальные биклики, поскольку они позволяют определять узкие места сети, основные магистрали, выполнять предобработку для разбиения исходной сети на подсети. Задача поиска кратчайшего пути в графе — хорошо известная задача теории графов, имеющая многие реальные приложения, такие как маршрутизация в коммуникационных сетях, логистическая оптимизация. Существует множество алгоритмов решения задачи маршрутизации без дополнительных ограничений на сеть [10, 13, 14]. Однако в случае, когда на сеть накладываются дополнительные ограничения, классические алгоритмы для построения кратчайших путей не применимы. Помимо этого, на производительность используемых алгоритмов существенное влияние оказывает увеличение размерности анализируемых графовых и гиперграфовых сетей. Это связано с тем, что классические алгоритмы либо не способны нахо-

дить точное решение поставленных задач, либо находят его за неприемлемое время [49, 51, 57]. В свою очередь, задача поиска всех максимальных биклик является труднорешаемой задачей, поскольку число биклик может экспоненциально зависеть от размеров исходной сети [16, 20, 72].

Задачи маршрутизации возникают в дорожных и телекоммуникационных сетях, при этом часто приходится иметь дело с различными расширениями задачи о кратчайшем пути, которые могут значительно повысить сложность решения [12, 40, 47, 51, 77, 81, 86]. Более того, в дорожных сетях логистические задачи могут решаться для отдельного района, города, края или целой страны. Большинство современных телекоммуникационных сетей являются мультисервисными, поскольку предоставляют пользователям множество разнообразных услуг, касающихся проводной телефонии, сотовой связи, кабельного телевидения и передачи данных. Качество обслуживания (Quality of Service, QoS) в мультисервисных сетях определяется параметрами, такими как полоса пропускания, задержка, вариация задержки, стоимость и надежность передачи данных [78, 86]. Таким образом, задача анализа качества обслуживания в телекоммуникационной сети может моделироваться задачей поиска кратчайших путей в ресурсоограниченной сети, где всякий запрос пользователя на оказание услуги предполагает определение некоторого (желательно оптимального по затратам, например, времени или стоимости) маршрута между двумя узлами с учетом ресурсных возможностей этой сети. Задачи перечислительного типа используются в дорожных и телекоммуникационных сетях в основном для анализа структурных элементов или выделения подсетей [20]. Задача моделирования адресного пространства в телекоммуникационных сетях связана с задачей поиска максимальных индуцированных биклик [59]. Например, максимальная индуцированная биклика может определять подсеть с топологией «звезда». Для анализа структуры дорожной сети можно использовать задачу перечисления всех максимальных индуцированных биклик.

В анализе семантических сетей востребованы задачи перечислительного типа, поскольку нахождение всех возможных конфигураций позволяет выделять скрытые знания и взаимосвязи из таких сетей [3, 55, 71]. Учитывая, что

подобные сети представляются как гиперграф, так и $(0,1)$-матрица, то поиск различных конфигураций можно осуществлять методами теории матриц и теории гиперграфов. Так, наиболее часто в анализе семантических сетей исследуются максимально полные подматрицы $(0,1)$-матрицы и максимальные индуцированные биклики. При этом задача выделения всех максимальных индуцированных биклик гиперграфа сводится к задаче построения конфигураций — полных единичных подматриц. Заметим, что задачи поиска таких конфигураций являются $\#P$-полными [16, 72].

В настоящее время активно разрабатываются новые, модифицируются известные алгоритмы для решения задач анализа, маршрутизации и проектирования дорожных, телекоммуникационных и семантических сетей, а также исследуются подходы, повышающие производительность существующих алгоритмов.

Степень разработанности темы исследования.

В задачу анализа сетей входит определение эффективности сети, нахождение узких мест, решение задачи о кратчайших путях, выделение подсетей и основных магистралей и др. Предъявление дополнительных ограничений к анализу сетей влечет за собой решение расширений задачи о кратчайшем пути, а также решение задачи определения различных конфигураций в сети.

Задача о кратчайшем пути в нестационарной сети была исследована в работе K. L. Cooke, E. Halsey [43]. В данной работе предложена модель поиска оптимального времени прибытия. В работе S. E. Dreyfus [49] рассмотрена возможность применения классического алгоритма Дейкстры для нахождения кратчайшего пути в нестационарной сети. Дальнейшие исследования включают усовершенствование алгоритма Дейкстры, выполнение предобработки исходной сети, использование новых подходов к решению задачи о кратчайшем пути в нестационарной сети. Значительный вклад в это направление внесли H. D. Sherali [81], D. Wagner [87], M. M. Nejad [77], D. Delling [46], Э. Х. Гимади [8].

Задача о кратчайшем пути в ресурсоограниченной сети является расширением классической задачи о кратчайшем пути с дополнительными ресурс-

ными функциями для каждой дуги. Задача о ресурсоограниченном кратчайшем пути была рассмотрена в работе H. Joksch [66]. В данной работе задача рассматривалась в терминах целочисленного линейного программирования и теоретико-графовой постановке, для которой был предложен алгоритм, основанный на методе динамического программирования. В работе G. Handler, I. Zang [60] был предложен двойственный алгоритм решения для задачи о ресурсоограниченном кратчайшем пути в терминах целочисленного линейного программирования. Значительный вклад в развитие подходов к решению задачи о кратчайшем пути в ресурсоограниченной сети внесли M. Dror [50], I. Dumitrescu [51], K. Mehlhorn [75], M. Jepsen [65], M. Horvarth [63], R. Hassin [61], G. Xue [88], W. Zhang [89].

Задачи поиска максимально полных подматриц и максимальных биклик являются задачами перечисления и поиска конфигураций в сети. Такие задачи применяются во многих прикладных областях. Поиск всех максимально полных подматриц используется в анализе формальных понятий. Его основные идеи и сведения задачи поиска формальных понятий к задаче поиска максимально полных подматриц были введены в следующих работах [16, 55, 72]. Развитие подходов к решению задачи активно исследуется российскими учеными С. О. Кузнецовым, С. А. Объедковым, Д. И. Игнатовым [16, 70–72], В. В. Быковой, Ч. М. Монгуш [3, 76]. Задача поиска максимальных индуцированных биклик связана с поиском максимально полных подматриц. Практическое применение максимальных индуцированных биклик в телекоммуникационных сетях исследовано в работе R. L. Graham, H. O. Pollak [59]. В других областях знаний максимальные биклики исследовались Е. В. Константиновой [69], Инной и Игорем Зверович [92], В. Б. Поповым [20].

Все рассматриваемые в работе задачи в большинстве случаев решаются применительно, к сетям представленным графами или гиперграфами, поэтому эти задачи носят комбинаторный характер. Развитие дорожных и телекоммуникационных сетей влечет за собой увеличение их сложности и размерности. Вследствие чего актуальна разработка новых и усовершенствование существующих алгоритмов комбинаторной оптимизации для данных задач, способных

находить решение за реальное время. Именно данное направление исследуется в настоящей диссертационной работе.

Цель и задачи исследования. Целью диссертационной работы является разработка алгоритмов комбинаторной оптимизации для анализа графовых и гиперграфовых сетей.

Для достижения цели были поставлены и решены следующие задачи.

1. Разработать и теоретически обосновать модификацию алгоритма ALT решения задачи поиска кратчайшего пути в нестационарной метрической сети, удовлетворяющей условию FIFO.

2. Разработать и теоретически обосновать приближенный алгоритм решения задачи поиска ресурсоограниченного кратчайшего пути в графе с одним ресурсом.

3. Разработать и теоретически обосновать алгоритм решения задачи поиска всех максимальных индуцированных биклик для графов и гиперграфов.

4. Создать комплекс программ, реализующий разработанные алгоритмы, для проверки результативности алгоритмов на случайных графах и гиперграфах и на реальных данных применительно к дорожным сетям.

Научная новизна результатов, представленных в диссертации.

1. Разработана модификация алгоритма ALT для задачи поиска кратчайшего пути в нестационарной метрической сети, удовлетворяющей условию FIFO. В отличие от классического алгоритма ALT, в модификации используется новая графовая модель, в которой всякая дуга сети определяется статическим и динамическим весами, интерпретируемыми как расстояние и скорость в сети. Предложенная модификация алгоритма ALT позволяет находить точное решение задачи поиска кратчайшего пути в нестационарной метрической сети, удовлетворяющей условию FIFO.

2. Разработан новый алгоритм RevTree приближенного решения задачи поиска ресурсоограниченного кратчайшего пути в сети с одним ресурсом. Алгоритм отличается от ранее существующих алгоритмов тем, что позволяет получить оценку точности решения на основе параметров исходной сети.

3. Разработан новый алгоритм HFindMIB решения задачи поиска всех

максимальных индуцированных биклик. Алгоритм отличается от ранее существующих алгоритмов тем, что в процессе генерации биклик использует новый гиперграфовый подход. Алгоритм находит и перечисляет в лексикографическом порядке все максимальные индуцированные биклики в графовых и гиперграфовых сетях.

Методология и методы исследования. Для решения поставленных в диссертационной работе задач был применен аппарат теории графов и гиперграфов, теории алгоритмов, комбинаторной оптимизации, а также методы объектно-ориентированного программирования. Комплекс программ для предложенных алгоритмов реализован на языке C++.

Теоретическая значимость работы. Результаты диссертации представляют теоретический интерес и вносят заметный вклад в изучение задач маршрутизации и задач перечислительного типа для графов и гиперграфов. Новые представления графовой модели и потенциальных функций для задачи о кратчайшем пути в нестационарной сети с условием FIFO могут быть использованы для моделирования других нестационарных сетей с различными ограничениями. Предложенный алгоритм для нахождения ресурсоограниченного пути может быть расширен на случай многих ресурсов. Метод оценки полученного решения может быть развит в общий подход для задач маршрутизации с несколькими весовыми функциями. Подход генерации максимальных индуцированных биклик может использоваться для развития методов генерации k-дольных клик гиперграфа.

Практическая значимость работы. Разработанные в диссертационной работе алгоритмы используются в Федеральном государственном автономном образовательном учреждении высшего образования «Сибирский федеральный университет» при подготовке бакалавров по специальностям 01.03.01 — Математика, 01.03.02 — Прикладная математика и информатика, 02.03.01 — Математика и компьютерные науки, при изучении дисциплин «Комбинаторные алгоритмы», «Технологии хранения и обработки больших данных», а также в научно-исследовательской работе магистрантов по направлению подготовки 01.04.02.06 — Прикладная математика и информатика в гуманитарных и

социально-экономических науках.

Разработанные в работе алгоритмы и комплекс программ ориентированы на эффективное применение в различных прикладных областях. В работе это продемонстрировано на примере реальных дорожных сетей городов, в частности, при интеллектуальном анализе дорожного графа города Красноярска в условиях ограничений. Применение алгоритмов к модельной дорожной сети, в которой добавлены новые дороги или перекрыты существующие, позволяет анализировать последствия таких решений путем определения наиболее востребованных дорог. Это может повысить качество работы алгоритмов выбора стратегий управления и дальнейшей оптимизации планов координаций светофоров на отдельных участках дорожной сети.

Соответствие паспорту специальности. Диссертационная работа соответствует области исследования специальности 05.13.17 — Теоретические основы информатики по п. 10 «Разработка основ математической теории языков и грамматик, теории конечных автоматов и теории графов» (пункты 1–3 научной новизны), по п. 5 «Разработка и исследование моделей и алгоритмов анализа данных, обнаружения закономерностей в данных и их извлечениях, разработка и исследование методов и алгоритмов анализа текста, устной речи и изображений» (пункт 3 научной новизны).

Положения и результаты, выносимые на защиту.

1. Модифицированный алгоритм ALT для решения задачи поиска кратчайшего пути в нестационарной метрической сети, удовлетворяющей условию FIFO, и оценки его сложности по времени.

2. Алгоритм RevTree для решения задачи поиска кратчайшего пути в ресурсоограниченной сети с одним ресурсом, доказательство корректности и оценка его сложности по времени и памяти.

3. Алгоритм HFindMCS поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа и оценка его сложности по времени.

4. Теорема об эквивалентности индуцированных двудольных подгиперграфов гиперграфа и двудольных подграфов соответствующего вершинного гра-

фа гиперграфа. Алгоритм HFINDMIB для поиска всех максимальных индуцированных биклик гиперграфа, основанный на этой теореме, и оценка его сложности по времени.

5. Комплекс программ для проверки результативности предложенных алгоритмов на случайных графах и гиперграфах и на реальных данных применительно к дорожным сетям.

Степень достоверности и апробация результатов работы. Достоверность результатов работы подтверждается строгими математическими доказательствами основных положений, экспериментальной проверкой результатов, численных расчетов на реальных данных и практической эффективности программных реализаций.

Основные результаты работы докладывались и обсуждались на Международной конференции Discrete Optimization and Operations Research DOOR-2016 (Владивосток, 2016), VII Международной конференции «Проблемы оптимизации и их приложения» (Омск, 2018), XVII Международной конференции имени А.Ф. Терпугова «Информационные технологии и математическое моделирование» ИТММ'18 (Томск, 2018), International Scientific Multi-Conference on Industrial Engineering and Modern Technologies FarEastCon'18 (Владивосток, 2018), 17 Всероссийской конференции с международным участием «Компьютерная безопасность и криптография» SIBECRYPT'18 (Абакан, 2018), 18 Всероссийской конференции с международным участием «Компьютерная безопасность и криптография» SIBECRYPT'19 (Томск, 2019), VII Международной конференции «Математика, её приложения и математическое образование» (Улан-Удэ, 2020), XIX Международной конференции имени А.Ф. Терпугова «Информационные технологии и математическое моделирование» ИТММ'20 (Томск, 2020), Семнадцатой Международной Азиатской школе-семинаре «Проблемы оптимизации сложных систем» OPCS'21 (Online, 2021), 24th International Conference on Distributed Computer and Communication Networks: Control, Computation, Communications DCCN'21 (Moscow, Online, 2021), Международной конференции «Вопросы логистики, управления и эксплуатации в транспортном коридоре Восток-Запад» PLMO'21 (Баку, 2021), научных

семинарах кафедры высшей и прикладной математики Сибирского федерального университета.

Работа поддержана Красноярским математическим центром, финансируемым Минобрнауки РФ в рамках мероприятий по созданию и развитию региональных НОМЦ (Соглашение 075-02-2020-1534/1).

Личный вклад автора. Постановка задач, представленных в диссертации, была сделана автором совместно с научными руководителями д.ф.-м.н, профессором В.В. Быковой и к.ф.-м.н, доцентом Д.В. Семеновой. Основные результаты, составляющие новизну диссертационной работы, получены лично автором. Обсуждение подходов, алгоритмов, результатов вычислительных экспериментов и подготовка публикаций осуществлялись совместно с научными руководителями. Автор самостоятельно выполнил разработку программных продуктов для созданных в диссертационной работе алгоритмов и получил соответствующие свидетельства о их государственной регистрации.

Публикации. По тематике диссертации опубликовано **22** работы, из них **5** статей в журналах, включенных в Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук (в том числе **4** статьи в российских научных журналах, индексируемых Web of Science и Scopus) [7, 23, 28, 83, 84], **3** свидетельства о государственной регистрации программы для ЭВМ [32–34], **14** публикаций в сборниках материалов международных и всероссийских научных и научно-практических конференций (в том числе **3** статьи в изданиях, индексируемых Web of Science и Scopus) [2, 4–6, 22, 24–27, 29–31, 82, 85].

Структура и объем диссертации. Диссертация состоит из введения, четырех глав, заключения, списка литературы, списка таблиц, списка иллюстраций, двух приложений. Общий объем диссертации составляет 95 страниц; иллюстративный материал представлен 15 рисунками и 14 таблицами; список литературы содержит 92 наименования.

Во **введении** к диссертации дается описание работы, раскрывается актуальность темы исследования, приводится обзор работ других авторов по изу-

чаемой тематике, формулируются цели и задачи исследования, излагается методология исследования, обосновывается теоретическая и практическая значимость диссертации, а также научная новизна результатов исследования.

В **первой главе** рассматривается задача о кратчайшем пути в нестационарной метрической сети с условием FIFO. Основным результатом первой главы является модифицированный алгоритм ALT, для которого предложены новые потенциальные функции и эвристика AdaHeuris для периодического обновления множества ориентиров. Для нестационарной метрической сети с условием FIFO доказана теорема о достаточном условии допустимости и преемственности потенциальной функции. Достаточное условие в теореме выражено неравенством треугольника. Сформулирована и доказана теорема о теоретической оценке сложности модифицированного алгоритма ALT. Полученная оценка сопоставима со сложностью задачи о кратчайшем пути в нестационарной метрической сети с условием FIFO и с другими подходами к ее решению.

Во **второй главе** рассматривается задача поиска кратчайшего пути в ресурсоограниченной сети. Основным результатом второй главы является приближенный алгоритм RevTree нахождения кратчайшего пути в сети с одним ресурсом. Точность алгоритма RevTree представлена теоремой и зависит от весовой и ресурсной функций дуг сети. Сформулирована и доказана теорема о теоретической оценке сложности алгоритма RevTree. Полученная оценка, с учетом точности алгоритма, сопоставима со сложностью задачи RCSP с одним ресурсом.

В **третьей главе** рассматриваются задачи перечислительного типа для сетей, представленных гиперграфом. Основным результатом главы являются алгоритмы HFindMCS и HFindMIB. Алгоритм HFindMCS решает задачу поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа, а алгоритм HFindMIB решает задачу перечисления всех максимальных индуцированных биклик гиперграфа. Сформулирована и доказана теорема об эквивалентности индуцированных двудольных подгиперграфов гиперграфа и двудольных подграфов вершинного графа. Данная теорема исполь-

зуется в алгоритме HFINDMIB. Для алгоритмов HFINDMCS и HFINDMIB сформулированы и доказаны теоремы о теоретической сложности и корректности.

В **четвертой главе** разработан комплекс программ, реализующий предложенные алгоритмы. Алгоритмы были протестированы на случайных и реальных сетях, представленных графами и гиперграфами. Приводится подробный анализ дорожных сетей на основе модифицированного алгоритма ALT и алгоритмов REVTREE и HFINDMIB.

В **заключении** диссертации сформулированы основные результаты и выводы, полученные на основе настоящей диссертационной работы.

Приложения содержат акты об использовании результатов диссертации.

Глава 1. Поиск кратчайшего пути в нестационарной метрической сети

Задача о кратчайшем пути в нестационарной сети состоит в поиске пути наименьшего веса в случае, когда веса дуг зависят от времени. Предполагается, что прохождение по всякой дуге требует некоторого времени. Это может повлечь за собой изменение весов дуг сети, описываемых функциями прибытия. Известно, что поиск кратчайшего пути в нестационарной сети общего вида без каких-либо ограничений на топологию сети и функции прибытия является NP-трудной задачей [81]. В случае, когда функции прибытия удовлетворяют условию FIFO (First-In First-Out), задача является полиномиально разрешимой [8, 68].

Данная глава посвящена задаче о кратчайшем пути для нестационарной метрической сети с условием FIFO. Для решения данной задачи предлагается модифицированный алгоритм ALT. Это соответсвует задаче 1 диссертационного исследования. Основные результаты опубликованы в работах [2, 4, 7, 23, 27, 29]. В параграфе 1.1 приведены определения, необходимые для постановки задачи. Задача рассматривается для метрического случая — веса дуг исходной сети удовлетворяют неравенству треугольника. Новый метод представления весов дуг гарантирует выполнение неравенства треугольника, что необходимо для корректной работы алгоритма ALT. В параграфе 1.2 выполнен обзор известных подходов к решению задачи о кратчайшем пути в нестационарной метрической сети. В параграфе 1.3 для нестационарной метрической сети с условием FIFO разработана модификация алгоритма ALT. Для нестационарной метрической сети с условием FIFO предложены специальные потенциальные функции, необходимые для работы алгоритма ALT. Доказана теорема о достаточном условии допустимости и преемственности таких потенциальных функций. Допустимость и преемственность потенциальных функций являются обязательными условиями для корректной работы алгоритма ALT. Также для алгоритма ALT предложена модификация — эвристика AdaHeuris для первой фазы расстановки ориентиров, которая выполняет периодическое обновление множества ориентиров и потенциальных функций. Сформулирована и доказана теорема о сложности модифицированного алгоритма ALT.

1.1 Постановка задачи о кратчайшем пути в нестационарной метрической сети с условием FIFO

Пусть задан ориентированный граф $G = (V, E)$ без кратных дуг и петель (далее просто граф), где V — множество вершин графа, а E — множество дуг графа.

В работе используются следующие обозначения и предположения:

(П1) величина $l_{xy} \geq 0$ является постоянной и может интерпретироваться как расстояние между вершинами $x, y \in V$;

(П2) расстояния между вершинами графа $G = (V, E)$ подчиняются неравенству треугольника;

(П3) величина $t > 0$, измеряется в условных единицах времени и принимает значения из некоторого конечного множества T;

(П4) функция $v_{xy}(t) > 0$ зависит от t и может интерпретироваться как скорость движения по дуге (x, y).

На основе предположений (П1)–(П4) введем следующее определение весовой функции.

Определение 1.1. Для всякой дуги $(x, y) \in E$ весовая функция $w_{xy}(t)$ есть отношение величин l_{xy} и $v_{xy}(t)$

$$w_{xy}(t) = \frac{l_{xy}}{v_{xy}(t)}. \tag{1.1}$$

Весовая функция $w_{xy}(t) \geq 0$ отображает время передвижения из вершины x в вершину y, при условии, что старт из вершины x осуществлен в момент времени t. Отметим, что функции $w_{xy}(t)$, $v_{xy}(t)$ зависят от t и, как следствие, являются дискретными с конечным множеством значений.

Последующие определения теории графов являются классическими для нестационарной метрической сети [8, 10, 13].

Сопоставим дуге $(x, y) \in E$ функцию прибытия $F_{xy}(t) = t + w_{xy}(t)$, где t — время отправления из вершины x, а $F_{xy}(t)$ — время прибытия в вершину y, при движении по дуге (x, y). Поскольку $t > 0$, $w_{xy}(t) \geq 0$, то всегда $F_{xy}(t) \geq t > 0$. Данное неравенство отражает непосредственный ход времени: «отправляясь

из вершины x в момент времени t невозможно прибыть в вершину y раньше времени t вне зависимости от значений l_{xy} и $v_{xy}(t)$».

Определение 1.2. Если для любых моментов времени $0 < t_1 \leq t_2$ выполняется неравенство

$$F_{xy}(t_1) \leq F_{xy}(t_2), \tag{1.2}$$

то говорят, что функция прибытия для дуги (x, y) является монотонной.

Определение 1.3. Граф $G = (V, E)$ с весовыми функциями (1.1) и монотонными функциями прибытия (1.2) будем называть нестационарной метрической сетью, удовлетворяющей условию FIFO.

Весовая функция $w_{xy}(t)$, удовлетворяющая условию FIFO представлена на рис. 1.1.

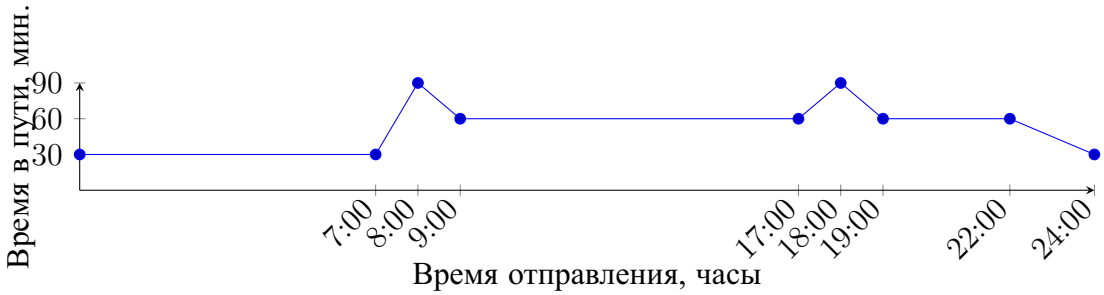


Рисунок 1.1 — Пример весовой функции $w_{xy}(t)$ для нестационарной метрической сети, удовлетворяющей условию FIFO

Говорят, что последовательность вершин $P = \{x_0, x_1, \ldots, x_k\}$ задает (s, d)-путь P из вершины s в вершину d в графе $G = (V, E)$, где $s = x_0$ — начало пути, $d = x_k$ — конец пути, $(x_{i-1}, x_i) \in E$ для $i = \overline{1, k}$. Если в G существует (s, d)-путь P, то вершина d достижима из s по пути P, что записывается как $s \overset{P}{\rightsquigarrow} d$. В случае, когда старт (s, d)-пути осуществлен в момент времени t_s путь обозначается как (P, t_s). Вес пути (P, t_s) вычисляется по формуле

$$w\left(P, t_s\right) = t_s + \sum_{i=0}^{k-1} w_{x_i x_{i+1}}\left(t_i\right), \tag{1.3}$$

где $t_0 = t_s$, $t_{i+1} = F_{x_i x_{i+1}}\left(t_i\right)$.

Путь (P, t_s) будем называть кратчайшим и обозначать как (s, d, t_s)-путь, если его вес равен

$$dist(s, d, t_s) = \min_P \left\{ w(P, t_s) \colon s \overset{P}{\rightsquigarrow} d \right\}. \tag{1.4}$$

Заметим, что значения величин $w(P, t_s)$ и $dist(s, d, t_s)$ всегда могут быть вычислены по формулам (1.3) и (1.4), если вершина d достижима из вершины s в графе G. Разница между $w(P, t_s)$ и $dist(s, d, t_s)$ представлена на рис. 1.2.

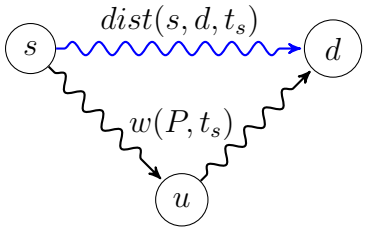


Рисунок 1.2 — Время прибытия при следовании по кратчайшему пути соответствует величине $dist(s, d, t_s)$, а при прохождении произвольного пути P время прибытия соответствует величине $w(P, t_s)$

Величину $w(P, t_s)$ будем трактовать как время прибытия в вершину d при прохождении (P, t_s)-пути, а величину $dist(s, d, t_s)$ — как самое раннее время прибытия в d при условии, что отправление из вершины s осуществлено в момент времени t_s. Тройку величин (s, d, t_s), где s и d — стартовая и целевая вершины соответственно, а t_s — стартовое время, будем называть запросом на поиск в нестационарной сети $G = (V, E)$ кратчайшего (s, d, t_s)-пути, или кратко (s, d, t_s)-запросом.

С использованием введенных выше понятий и обозначений задача поиска кратчайшего пути в нестационарной метрической сети (Time-Dependent Shortest Path problem, TDSP) с условием FIFO формулируется следующим образом.

Заданы: нестационарная метрическая сеть $G = (V, E)$ с условием FIFO, (s, d, t_s)-запрос.

Требуется: найти значение $dist(s, d, t_s)$ для кратчайшего (s, d, t_s)-пути и последовательность вершин, образующих этот путь.

Задача TDSP всегда имеет решение, возможно не единственное, если вершина d достижима из вершины s в графе G. Неотрицательность весов дуг сети и условие FIFO (1.2) требуются для корректной работы известного алгоритма Дейкстры, а также его различных версий, включая алгоритмы A* и ALT [8]. Кроме того, условие FIFO обеспечивает полиномиальную разрешимость этой задачи [8, 68]. Задача TDSP активно используется в телекоммуникационных сетях для маршрутизации пакетов данных в загруженных участках сети, при моделировании дорожных ситуаций в транспортных сетях.

1.2 Обзор подходов к решению задачи TDSP

Задача поиска кратчайшего пути в нестационарной метрической сети с уловием FIFO, является расширением классической задачи поиска кратчайшего пути в графе. Известно много алгоритмов для поиска кратчайших путей в графе [8, 13, 21], некоторые из них адаптированы для нестационарных сетей [45, 46, 77, 87].

Классический алгоритм Дейкстры способен решать задачу поиска кратчайшего пути в нестационарной метрической сети с условием FIFO. Алгоритм Дейкстры равномерно расширяет пространство поиска решения в исходном графе $G = (V, E)$, начиная от стартовой вершины и далее до целевой вершины. Данный алгоритм требует $\mathcal{O}\left(|V|^2\right)$ времени для точного решения задачи TDSP [8].

При разработке современных систем маршрутизации приходится сталкиваться с проблемой размерности сетей. Так, сеть может содержать несколько десятков тысяч или сотен тысяч узлов, и требуется найти за несколько секунд оптимальный маршрут между узлами этой сети. В подобных условиях классические алгоритмы поиска кратчайших путей, включая алгоритм Дейкстры, работают неприемлемо долго. Чтобы справиться с этой проблемой используются различные приемы ускорения классических алгоритмов [21]. Большинство этих приемов основаны на ускорении алгоритма Дейкстры путем применения двухфазного подхода к решению задачи TDSP. Первая фаза такого алгоритма — предобработка исходного графа $G = (V, E)$, вторая фаза — выполнение

(s, d, t_s)-запроса в графе, полученном после предобработки. Предобработка сводится к просмотру графа $G = (V, E)$ и анализу его структуры с целью извлечения информации, позволяющей ускорить вторую фазу алгоритма.

Известны различные реализации двухфазного подхода алгоритма Дейкстры [87]. Их принято разделять на следующие основные группы (табл. 1.1): иерархические алгоритмы [77], алгоритмы маркировки [46], алгоритмы маршрутизации по ориентирам [57, 58].

Таблица 1.1 — Подходы к решению задачи TDSP

Подход	Представитель	Особенность
Иерархический	HTNGD	Итеративное представление частей графа в виде одной вершины
Маркировочный	Arc-flags	Разбиение исходного графа на части, маркировка связующих ребер
Маршрутизация по ориентирам	ALT	Выделение специальных вершин графа, для вычисления потенциальной функции

Иерархические алгоритмы на фазе предобработки преобразуют исходный граф в конечное число связанных между собой вспомогательных графов — уровней. Уровни создаются таким образом, чтобы размерность каждого следующего уровня была меньше предыдущего. Связи между уровнями устанавливаются через представление одной вершиной текущего уровня некоторого множества вершин предыдущего уровня. Технология выделения подобных множеств вершин и формирование отдельных уровней определяется посвоему в каждом конкретном иерархическом алгоритме [87]. Однако на фазе выполнения (s, d, t_s)-запроса иерархические алгоритмы, как правило, ведут себя одинаково. Они последовательно запускают алгоритм Дейкстры для каждого отдельного уровня, определяя в нем область поиска для последующих уровней. Тем самым, область поиска кратчайшего (s, d, t_s)-пути сужается. Степень ускорения иерархических алгоритмов в сравнении с алгоритмом Дейкстры во многом зависит от того, как и сколько уровней выделено в исходном графе. Современным представителем данной группы алгоритмов является алгоритм HTNGD [77].

Алгоритм Arc-flags относится к алгоритмам маркировки. Функционирование данного алгоритма подобно работе иерархических алгоритмов. Алгоритм Arc-flags на первой фазе разбивает исходный граф на несколько примерно равных частей, затем, просматривая все ребра графа, выделяет среди них ребра, лежащие на стыке образованных частей графа. Выделенное множество ребер алгоритм использует для создания меток, приписываемых всем ребрам графа. Именно эти метки играют роль «флажков» на маршруте движения и позволяют сузить пространство поиска алгоритма Дейкстры. Эффективность алгоритмов маркировки, включая алгоритм Arc-flags, зависит преимущественно от числа выделяемых частей графа и стратегии формирования меток [46].

В алгоритмах маршрутизации по ориентирам на фазе предобработки выбирается некоторое множество вершин — ориентиров, на основе которых вычисляются потенциальные функции. Потенциальные функции предназначены для сокращения пространства поиска кратчайшего (s, d, t_s)-пути. Степень ускорения алгоритмов маршрутизации по ориентирам в сравнении с алгоритмом Дейкстры главным образом зависит от числа ориентиров и стратегии их размещения в вершинах графа. Следует отметить, что однократная расстановка ориентиров требует гораздо меньше времени на предобработку исходного графа, чем предобработка, выполняемая иерархическими алгоритмами и алгоритмами маркировки, поскольку последние трансформируют структуру исходного графа. Это неизбежно влечет увеличение времени выполнения второй фазы алгоритма. Однако следует отметить, что потенциальные функции должны обладать специальными свойствами, в противном случае алгоритм будет находить некорректное решение задачи.

Все известные алгоритмы маршрутизации по ориентирам лишь однократно тем или иным способом расставляют ориентиры в вершинах исходного графа. Очевидно, что повышение эффективности данных алгоритмов — это совершенствование процедуры расстановки ориентиров, включая многократность ее применения ко всему набору ориентиров или некоторой его части. В системах маршрутизации, функционирующих в режиме онлайн и выполняющих различные (s, d, t_s)-запросы, для этих целей может быть использована

история обработки этих запросов. Различные стратегии выбора хранимой информации об истории обработки предшествующих запросов и использования этой информации порождают разнообразие новых алгоритмов маршрутизации по ориентирам. Современным представителем данной группы алгоритмов является алгоритм ALT.

Алгоритм ALT (A* with Landmarks & Triangle) представляет собой версию алгоритма A*, осуществляющую целенаправленный поиск по ориентирам. При этом под ориентиром понимается некоторая выделенная вершина исходного графа. Алгоритм A* — эквивалент алгоритма Дейкстры, работающий с потенциальными функциями [21, 58]. Алгоритм ALT состоит из двух фаз: на первой фазе выполняется расстановка ориентиров и вычисление на их основе потенциальных функций, а на второй фазе с помощью алгоритма A* осуществляется выполнение (s, d, t_s)-запроса. Роль потенциальных функций — оценка снизу значения кратчайшего пути от текущей вершины до целевой вершины. Напомним, что традиционно в алгоритме Дейкстры применяется лишь верхняя оценка пути от стартовой вершины до текущей вершины. Использование в алгоритме A* именно этих двух оценок определяет наиболее «оптимистичное направление» движения от текущей вершины к целевой вершине искомого пути.

Известно, что алгоритм A* работает корректно, т. е. находит точное решение задачи о кратчайшем пути в графе за время $\mathcal{O}(|V|^2)$, если потенциальные функции обладают свойствами допустимости и преемственности [21, 58]. Для графа с постоянными неотрицательными весовыми функциями данные свойства всегда выполнимы, если веса дуг исходного графа подчиняются неравенству треугольника [58].

На сегодняшний день все описанные выше приемы ускорения алгоритма Дейкстры востребованы практикой и продолжают совершенствоваться. Область применения каждого из них определяется спецификой сети, в которой осуществляется маршрутизация. Здесь, как правило, учитываются такие факторы как протяженность и плотность сети, реальность или виртуальность основных (магистральных) маршрутов сети [58, 81, 87].

В диссертации предлагается модификация алгоритма ALT с усовершенствованной процедурой расстановки ориентиров, а также сформулировано и доказано необходимое условие для потенциальных функции. Усовершенствование состоит в применении адаптивной эвристики, которая многократно корректирует текущий набор ориентиров на основе истории предыдущих запросов с целью наиболее эффективного исполнения текущего запроса. Заметим, что все версии алгоритма Дейкстры, включая алгоритм ALT, находят точное решение задачи TDSP в указанной выше постановке за полиномиальное время [45, 46, 77].

1.3 Модифицированный алгоритм ALT

Определим неравенство треугольника и потенциальные функции для нестационарной сети.

Определение 1.4. Максимальная скорость, с которой можно двигаться по всякой дуге исходного графа, определяется величиной

$$v_{\max} = \max_{(x,y)\in E} \left\{ \max_{t\in T} [v_{xy}(t)] \right\} > 0. \tag{1.5}$$

Заметим, что величина $v_{\max}$ постоянна для заданного графа $G = (V, E)$ на рассматриваемом промежутке времени T. Вместо формулы (1.1) для каждой дуги $(x, y) \in E$ графа $G = (V, E)$ определим ее «оптимистичный» вес.

Определение 1.5. Вес дуги $(x, y) \in E$, при движении по ней с максимальной скоростью (1.5), определяется как

$$w_{xy}^{\nabla} = \frac{l_{xy}}{v_{\max}}. \tag{1.6}$$

Определение веса дуги (x, y) по формуле (1.6) приводит к тому, что величина w_{xy}^{∇} не зависит от t и соответствует времени передвижения от вершины x к вершине y при условии отсутствия каких-либо препятствий, приводящих к снижению скорости по этой дуге. Отсюда для всякой дуги $(x, y) \in E$ и любого $t \in T$ верна оценка

$$w_{xy}^{\nabla} \le w_{xy}(t).$$

Принимая во внимание формулу (1.3) и введенную оценку весов (1.6), вес пути P, идущего от вершины s к вершине d, будем формировать следующим образом:

$$w^{\nabla}(P) = \sum_{i=0}^{k-1} w^{\nabla}_{x_i x_{i+1}}.$$

В отличие от формулы (1.3) в данном случае вес пути P интерпретируем далее как время движения от стартовой вершины до целевой вершины этого пути. Таким образом, значение величины $w^{\nabla}(P)$ зависит только от «оптимистичных» весов дуг и не зависит от времени выхода из стартовой вершины. Очевидны оценки

$$w^{\nabla}(P) \le w(P, t_s),$$
$$t_s \le w(P, t_s) - w^{\nabla}(P).$$

Аналогично формуле (1.4) для кратчайшего (s, d, t_s)-пути имеем:

$$dist^{\nabla}(s, d) = \min_{P} \left\{ w^{\nabla}(P) \colon s \underset{P}{\rightsquigarrow} d \right\}. \tag{1.7}$$

Вес $dist(s, d, t_s)$ кратчайшего (s, d, t_s)-пути для любых s, d и t_s ограничен снизу значением $dist^{\nabla}(s, d)$:

$$dist^{\nabla}(s, d) \le dist(s, d, t_s). \tag{1.8}$$

Исходя из (1.7) и (1.8), для конкретно заданных s, d величина $dist^{\nabla}(s, d)$ — минимальное время, необходимое для передвижения в $G = (V, E)$ из вершины s в вершину d.

Используя величины, вычисляемые по формуле (1.7), определим неравенство треугольника для нестационарной сети следующим образом:

$$dist^{\nabla}(x, y) + dist^{\nabla}(y, z) \ge dist^{\nabla}(x, z), \tag{1.9}$$

где x, y, z произвольные вершины графа $G = (V, E)$. Если $G = (V, E)$ — нестационарная сеть, тогда данное неравенство означает, что минимальное время передвижения от вершины x в вершину z по кратчайшему пути всегда не больше времени движения в обход этого пути через некоторую вершину y. Неравенство треугольника представлено на рис. 1.3.

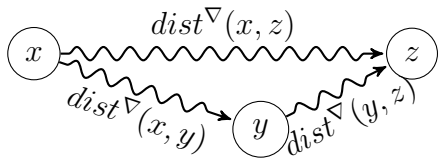


Рисунок 1.3 — Неравенство треугольника (1.9) для величин $dist^{\nabla}(x,z)$, $dist^{\nabla}(x,y)$, $dist^{\nabla}(y,z)$

Согласно предположениям (П1) и (П2), расстояния между вершинами графа G удовлетворяют неравенству треугольника. В этих условиях неравенство (1.9) всегда выполнимо, поскольку величины $dist^{\nabla}(x,y)$, $dist^{\nabla}(y,z)$, $dist^{\nabla}(x,z)$ определены через веса дуг, отмасштабированные с помощью формулы (1.6).

Определение 1.6. Пусть $L \subseteq V$ — непустое множество вершин сети $G = (V,E)$, в которых установлены ориентиры. Для каждой вершины $x \in V$ и всякого ориентира $l \in L$ определим следующие функции:

$$\pi_{l+}(x) = dist^{\nabla}(l,d) - dist^{\nabla}(l,x), \tag{1.10}$$

$$\pi_{l-}(x) = dist^{\nabla}(x,l) - dist^{\nabla}(d,l), \tag{1.11}$$

$$\pi_{l}(x) = \max\{0, \pi_{l+}(x), \pi_{l-}(x)\}. \tag{1.12}$$

Положительную величину

$$\pi_{L}(x) = \max_{l \in L} \pi_{l}(x) \tag{1.13}$$

назовем потенциальной функцией вершины x относительно множества ориентиров L.

Переход в формулах (1.10)–(1.12) от длин кратчайших путей, определяемых формулой (1.4), к их нижним оценкам (1.7) — особенность определения потенциальных функций для нестационарной сети (рис. 1.4).

Потенциальные функции, вычисленные по формулам (1.10)–(1.13), обладают свойствами допустимости и преемственности. Эти свойства гарантируют корректную работу алгоритма A*, т. е. нахождение им точного значения кратчайшего пути между двумя заданными вершинами графа. Опираясь на работу [21], определим данные свойства применительно к нестационарным сетям.

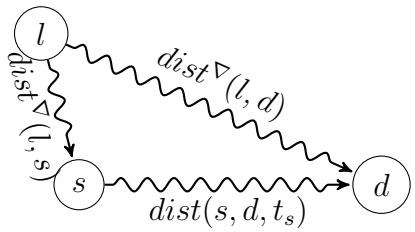


Рисунок 1.4 — Потенциальная функция $\pi_{l+}(s) = dist^{\nabla}(l,d) - dist^{\nabla}(l,s)$ отражает наилучшее время прохождения пути из вершины s в вершину d относительно ориентира l

Потенциальная функция, удовлетворяющая неравенствам

$$0 \leq \pi_L(x) \leq dist(x,d,t_x), \tag{1.14}$$

называется допустимой.

Потенциальная функция называется преемственной, если

$$\pi_L(s) \leq dist(s,x,t_s) + \pi_L(x). \tag{1.15}$$

Важно подчеркнуть, что для корректной работы алгоритма A* требуется выполнимость неравенств (1.14), (1.15) для любых $s,x,d \in V, t_s \in T$ и произвольного непустого множества ориентиров $L \subseteq V$.

Теорема 1.1 (Достаточное условие допустимости и преемственности потенциальной функции)**.** *Если для нестационарной сети $G = (V,E)$ выполняется неравенство треугольника*

$$dist^{\nabla}(x,y) + dist^{\nabla}(y,z) \geq dist^{\nabla}(x,z). \tag{1.16}$$

то потенциальная функция $\pi_L(x)$ вершины x относительно множества ориентиров L удовлетворяет условиям

– *допустимости* $\pi_L(x) \leq dist^{\nabla}(x,d) \leq dist(x,d,t_s)$*;*

– *преемственности* $\pi_L(s) \leq dist(s,x,t_s) + \pi_L(x)$.

Доказательство. Докажем допустимость потенциальных функций. Исходя из (1.8) и (1.9), для любых $x,d \in V, l \in L$ и $t_x \in T$ имеем

$$\pi_{l+}(x) = dist^{\nabla}(l,d) - dist^{\nabla}(l,x) \leq dist^{\nabla}(x,d) \leq dist(x,d,t_x),$$
$$\pi_{l-}(x) = dist^{\nabla}(x,l) - dist^{\nabla}(d,l) \leq dist^{\nabla}(x,d) \leq dist(x,d,t_x).$$

Поскольку всегда $0 \leq dist(x, d, t_x)$, то также

$$\pi_l(x) = \max\{0, \pi_{l+}(x), \pi_{l-}(x)\} \leq dist(x, d, t_x),$$
$$\pi_L(x) = \max_{l \in L} \pi_l(x) \leq dist(x, d, t_x),$$

т. е. неравенства (1.14) справедливы для любых $x, d \in V, t_x \in T$ и непустого множества ориентиров $L \subseteq V$.

Докажем преемственность потенциальных функций. Согласно (1.8) и (1.9) для всякого ориентира $l \in L$ и любых $s, x \in V, t_s \in T$ верны соотношения

$$dist^{\nabla}(s, x) \leq dist(s, x, t_s),$$
$$dist^{\nabla}(l, x) - dist^{\nabla}(l, s) \leq dist^{\nabla}(s, x).$$

Отсюда получаем

$$dist^{\nabla}(l, x) - dist^{\nabla}(l, s) \leq dist(s, x, t_s). \tag{1.17}$$

Добавим в обе части неравенства (1.17) положительную величину $dist^{\nabla}(l, d)$

$$dist^{\nabla}(l, d) + dist^{\nabla}(l, x) - dist^{\nabla}(l, s) \leq dist(s, x, t_s) + dist^{\nabla}(l, d).$$

Перенесем $dist^{\nabla}(l, x)$ из левой части этого неравенства в его правую часть

$$dist^{\nabla}(l, d) - dist^{\nabla}(l, s) \leq dist(s, x, t_s) + dist^{\nabla}(l, d) - dist^{\nabla}(l, x).$$

С учетом (1.10) данное неравенство принимает вид:

$$\pi_{l+}(s) \leq dist(s, x, t_s) + \pi_{l+}(x).$$

Аналогичным образом для ориентира $l \in L$ и формулы (1.11) получаем

$$\pi_{l-}(s) \leq dist(s, x, t_s) + \pi_{l-}(x).$$

Тогда согласно (1.12) для ориентира $l \in L$ при любых $s, x, d \in V$ и $t_s \in T$ верно неравенство

$$\pi_l(s) \leq dist(s, x, t_s) + \pi_l(x).$$

Значит, условие (1.15) всегда справедливо для любых $s, x, d \in V, t_s \in T$ и непустого множества ориентиров $L \subseteq V$. □

В данной работе будем использовать следующее определение нестационарной метрической сети.

Определение 1.7. Нестационарная сеть $G = (V, E)$, для которой неравенство (1.9) справедливо при любых $x, y, z \in V$ называется нестационарной метрической сетью.

Пусть задача TDSP решается для нестационарной сети $G = (V, E)$ применительно к последовательности σ, содержащей конечное число (s, d, t_s)-запросов. Предлагаемая в работе модификация алгоритма ALT состоит в применении потенциальных функций (1.10)–(1.13) и адаптивной эвристики расстановки ориентиров AdaHeuris. Схема алгоритма представлена на рис. 1.5.

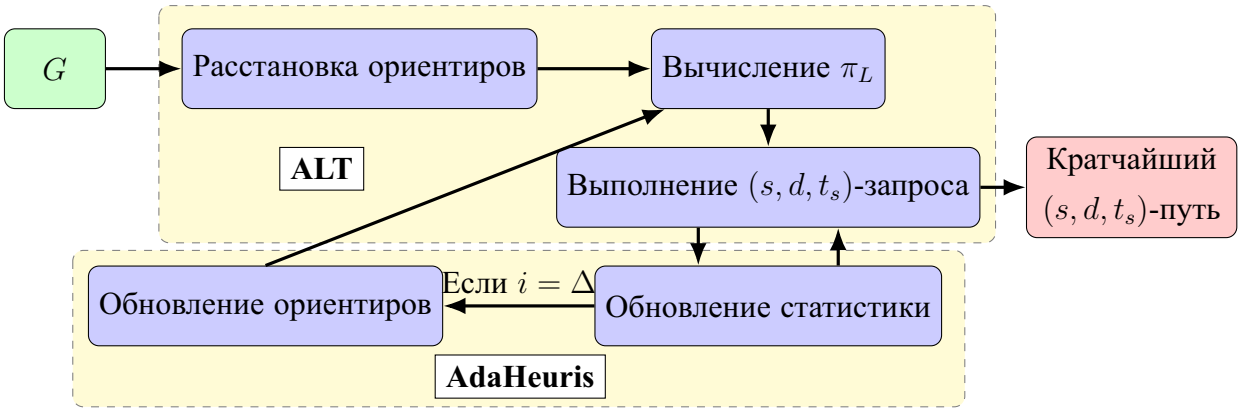


Рисунок 1.5 — Схема модифицированного алгоритма ALT

Суть эвристики AdaHeuris состоит в следующем. Перед выполнением первого (s, d, t_s)-запроса выполняется случайная расстановка заданного числа $K = |L|$ ориентиров в вершинах графа $G = (V, E)$ Далее выбранное множество ориентиров L периодически обновляется через каждые $\Delta > 0$ запросов. Полагается, что $|V| \gg K$ и $|\sigma| \gg \Delta$. При выполнении последовательности запросов σ накапливается информация о результативности текущих ориентиров. Всякий раз, когда ориентир предлагает наилучшую оценку пути в соотношении (1.13) среди всего набора ориентиров, он получает очко. Процесс обновления ориентиров осуществляется после завершения Δ-го запроса. Пусть l_{old} — ориентир с наименьшим количеством очков, набранных за пери-

од Δ, т. е. низкорезультативный. Данный ориентир подлежит перемещению в вершину x, удовлетворяющую двум условиям:

– вершина x в алгоритме A* получила временную метку при обработке предыдущих (s, d, t_s)-запросов и это состояние осталось неизменным на момент обновления ориентиров;
– наибольшая удаленность вершины x в смысле (1.7) от множества ориентиров $L \setminus \{l_{old}\}$ с учетом правила «тупого угла».

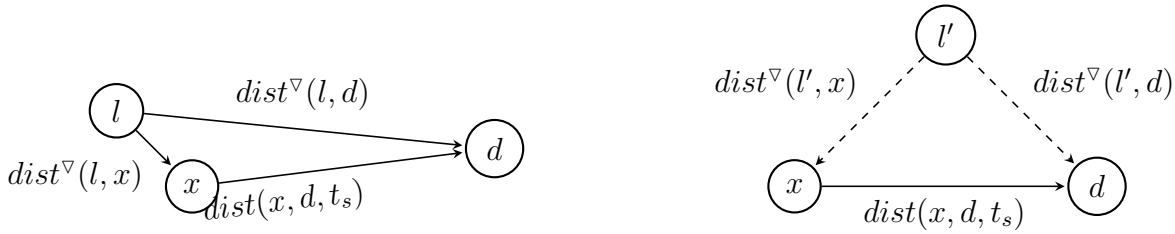
После такого обновления множества L информация об эффективности ориентиров начинает формироваться заново. Правило «тупого угла» обеспечивает наилучшую нижнюю оценку (1.13), поскольку согласно неравенству треугольника (1.9) и оценке (1.8) верны соотношения

$$dist(x, d, t_s) \geq dist^{\triangledown}(x, d) \geq dist^{\triangledown}(l, d) - dist^{\triangledown}(l, x) = \pi_l(x).$$

Различное положение ориентира l приводит к различным значениям разности

$$dist(x, d, t_s) - \pi_l(x) \geq 0.$$

Указанная разность тем меньше, чем больше угол при вершине x. Это хорошо иллюстрирует рис. 1.6.



а) б)

Рисунок 1.6 — а) расположение ориентира l отвечает правилу «тупого угла» и приводит к значению $dist^{\triangledown}(l, d) - dist^{\triangledown}(l, x) = \pi_l(x)$; б) для ориентира l' это правило не выполняется, что дает $dist^{\triangledown}(l', d) - dist^{\triangledown}(l', x) = \pi_{l'}(x) \leq \pi_l(x)$

Приведем описание эвристики AdaHeuris. В качестве исходных данных эвристика AdaHeuris использует нестационарную метрическую сеть $G = (V, E)$ с условием FIFO и число ориентиров K. Последовательность запросов σ и период Δ обновления множества ориентиров L применяются непосредственно в модифицированном алгоритме ALT.

Эвристика состоит из трех процедур, которые вызываются в соответствующих местах модифицированного алгоритма ALT:

– процедура *Инициализация* формирует вспомогательные массивы $Landmark$, $Stat$, $Passed$, $Space$ и выполняет первоначальную расстановку ориентиров случайным образом с помощью стандартной функции random;
– процедура *История обработки запросов* собирает сведения о результатах выполненного текущего (s, d, t_s)-запроса и заносит их в массивы $Passed$ и $Space$;
– процедура *Обновление ориентиров* выбирает низкорезультативный ориентир и перемещает его в вершину, выбор которой осуществляется в массиве $Space$ с учетом правила «тупого угла».

Описание этих процедур на псевдокоде представлено в алгоритмах 1.1–1.3.

Алгоритм 1.1. Процедура *Инициализация*

$Landmark \leftarrow \oslash$
$Stat \leftarrow \oslash$
$Passed \leftarrow \oslash$
$Space \leftarrow \oslash$
$i \leftarrow 0$
Для $i < K$
    $Landmark(i) \leftarrow$ random$(x) \in V$
    Для каждого $v \in V$
        Вычислить кратчайшие $(Landmark(i), v)$-пути и $(v, Landmark(i))$-пути
    Конец цикла
    $i \leftarrow i + 1$
Конец цикла

Алгоритм 1.2. Процедура *Обновление ориентиров*

Найти индекс l_{old}, такой что $Stat(l_{old}) = \min Stat(i) \colon i = 1, \ldots, K$
Вычислить номер вершины u, такой что
    $d_{mark}(new) = min(d_{mark}(i) \colon i \in Space)$
$Landmark(l_{old}) \leftarrow new$
Вычислить для вершины new и каждой вершины $x \in V$ кратчайшие (new, x)-пути и (x, new)-пути
$Stat \leftarrow \oslash$

Алгоритм 1.3. Процедура *История обработки запросов*

1: **Для** каждой вершины $x \in V$
2: **Если** x имеет временную метку и $x \notin Space$ и $x \notin Passed$, **то**
3: $Space \leftarrow x$
4: **Конец условия**
5: **Если** x имеет постоянною метку и $x \in Space$, **то**
6: **Удалить** x из $Space$
7: $Passed \leftarrow x$
8: **Конец условия**
9: **Конец цикла**

Использованные в эвристике ADAHEURIS вспомогательные массивы имеют следующие назначение. Массив $Landmark$ служит для формирования и хранения множества L, причем i-я строка этого массива содержит номер вершины, где установлен i-й ориентир, и длины всех кратчайших путей от этого ориентира до всех других вершин графа $G = (V, E)$, $i = 1, \ldots, K$. В массив $Stat$ записывается информация о результативности каждого ориентира из L. Массивы $Passed$ и $Space$ предназначены для хранения номеров вершин, получивших в алгоритме A* постоянные и временные метки при обработке предшествующих (s, d, t_s)-запросов. В данных массивах накапливается история обработки запросов — информация о множестве вершин, пройденных алгоритмом A* в процессе выполнения обработки части запросов из последовательности σ. Это множество вершин для алгоритма A* является пространством поиска решений, причем вершины из массива $Space$ рассматриваются в эвристике ADAHEURIS в качестве «границы» этого пространства, куда перемещаются низкорезультативные ориентиры. В худшем случае мощность пространства поиска решений для графа $G = (V, E)$ составляет $|V|$. Цель эвристики ADAHEURIS — уменьшение мощности этого пространства.

Кроме представленных выше процедур, эвристика ADAHEURIS содержит также процедуру *Вычисление потенциальной функции*, описанную в алгоритме 1.4.

Алгоритм 1.4. Процедура *Вычисление потенциальной функции*

1: **Вычислить** $\pi_L(x)$ для текущей вершины $x \in V$
2: **Найти** ориентир l, такой что $\pi_l(x) = \pi_L(x)$
3: $Stat(l) \leftarrow Stat(l) + 1$
4: **Возвратить** $\pi_L(x)$

Данная процедура вложена в алгоритм A*. С помощью этой процедуры выполняются следующие функции: вычисление потенциальной функции для текущей вершины $x \in V$; сбор информации о результативности ориентиров из L.

Модификация алгоритма ALT заключается в применении потенциальных функций (1.13) и предложенной эвристики AdaHeuris для расстановки, обновления ориентиров и формирования истории обработки запросов. Описание этих действий на псевдокоде представлено в алгоритме 1.5.

Алгоритм 1.5. Модифицированный алгоритм ALT

1: **Выполнить** процедуру *Инициализация*
2: $i \leftarrow 0$
3: **Для** каждого (s, d, t_s)-запроса из σ
4: **Выполнить** алгоритм A*
5: **Выполнить** процедуру *История обработки запросов*
6: $i \leftarrow i + 1$
7: **Если** $i = \Delta$, **то**
8: **Выполнить** процедуру *Обновление ориентиров*
9: $i \leftarrow 0$
10: **Конец условия**
11: **Конец цикла**

Входными данными для модифицированного алгоритма ALT являются:

- нестационарная метрическая сеть $G = (V, E)$, удовлетворяющая условию FIFO;
- последовательность (s, d, t_s)-запросов σ;
- число ориентиров K;
- период обновления ориентиров Δ.

Выходом модифицированного алгоритма ALT являются последовательность

вершин, образующих кратчайший путь, и длина этого пути для каждого отдельного (s, d, t_s)-запроса из σ.

Реализация алгоритма A* в модифицированном ALT отличается от традиционных его реализаций, описанных в [21, 58], следующим:

– вычисление потенциальных функций по формулам (1.10)–(1.13);
– сбор информации о результативности ориентиров, которая далее применяется при обновлении ориентиров.

Модифицированный алгоритм ALT является полиномиальным по времени, о чем свидетельствует следующая теорема.

Теорема 1.2 (О сложности модифицированного алгоритма ALT)**.** *Модифицированный алгоритм ALT находит точное решение задачи TDSP для последовательности σ, состоящей из конечного числа $(s,\ d,\ t_s)$-запросов, в графе $G = (V, E)$ за время $\mathcal{O}\left(|\sigma| \cdot |V|^2\right)$.*

Доказательство. Выполнение на шаге 1 процедуры *Инициализация*, в которой осуществляются первоначальная расстановка ориентиров и формирование массивов $Landmark$, $Stat$, $Passed$, $Space$, составляет $\mathcal{O}\left(|V|^2 \cdot K\right)$, поскольку только формирование массива $Landmark$ требует $\mathcal{O}\left(|V|^2 \cdot K\right)$ времени. Время работы алгоритма A*, выполняемого на шаге 4, традиционно составляет $\mathcal{O}\left(|V|^2\right)$. Отличительные особенности реализации алгоритма A* в модифицированном ALT, выраженные в алгоритме 4, не изменяют эту оценку. Время работы процедуры *История обработки запросов*, выполняемой на шаге 5, сопоставимо с мощностью пространства поиска решения и в худшем случае составляет $|V|$. Процедура *Обновление ориентиров*, выполняемая на шаге 8, осуществляет выбор низкорезультативного ориентира l_{old} и его перемещение в новую вершину new из массива $Space$, а также вычисление кратчайших путей от вершины new до всех других вершин графа. Для выполнения этих действий требуется $\mathcal{O}\left(|V|^2\right)$ времени. Таким образом, однократная реализация шагов 4–10 модифицированного алгоритма ALT требует $\mathcal{O}\left(|V|^2\right)$ времени. Поскольку число повторений цикла 3–11 совпадает с числом $|\sigma|$ запросов, входящих в последовательность σ, то время выполнения данного цикла и модифицированного алгоритма ALT в целом составляет $\mathcal{O}\left(|\sigma| \cdot |V|^2\right)$. □

Теоретическая оценка модифицированного алгоритма ALT, полученная в теореме 1.2, сопоставима с известными алгоритмами решения задачи TDSP [46, 57, 58, 77, 87]. Вычислительные эксперименты, представленные в главе 4, демонстрируют производительность предложенного алгоритма как на случайных, так и на реальных сетях.

1.4 Выводы по главе 1

Основным результатом данной главы является модифицированный алгоритм ALT для нахождения кратчайшего пути в нестационарной метрической сети с условием FIFO.

Модификация алгоритма ALT заключается в применении новых потенциальных функций $\pi_L(x)$ (определение 1.6) и эвристики AdaHeuris для периодического обновления множества ориентиров. Для нестационарной метрической сети с условием FIFO доказана теорема 1.1 о достаточном условии допустимости и преемственности потенциальной функции. Достаточное условие в теореме 1.1 выражено неравенством треугольника. Это условие зачастую выполнено для реальных сетей. Приведена теоретическая оценка модифицированного алгоритма ALT в теореме 1.2. Полученная оценка сопоставима со сложностью задачи TDSP и с другими подходами к ее решению.

Следует отметить следующие особенности модифицированного алгоритма ALT.

1. Если в модифицированном алгоритме ALT число ориентиров положить равным $|V|$, то время работы процедуры *Инициализация* составит $\mathcal{O}\left(|V|^3\right)$. В этой ситуации, когда ориентиры размещены во всех вершинах графа, время выполнения процедуры *Инициализация* превосходит время однократной реализации всех других шагов модифицированного алгоритма ALT. Это свидетельствует о том, что использование большого числа ориентиров в модифицированном алгоритме ALT затратно по времени. Поэтому для эффективной работы данного алгоритма целесообразно исходить из фиксированного числа ориентиров K при условии, что $|V| \gg K$.

2. При доказательстве теоремы 1.2 оценка времени работы модифициро-

ванного алгоритма ALT вычислялась применительно к худшему случаю для Δ, т. е. когда $\Delta = 1$. При $\Delta = 1$ обновление ориентиров происходит после выполнения каждого (s, d, t_s)-запроса, что затратно по времени и не всегда оправдано. Поэтому для эффективной работы модифицированного алгоритма ALT подбор значения Δ целесообразно осуществлять экспериментальным путем, исходя из типичного для рассматриваемой сети содержания запросов и их возможных последовательностей.

3. Известно, что алгоритм Дейкстры находит точное решение задачи TDSP для заданного (s, d, t_s)-запроса за время $\mathcal{O}\left(|V|^2\right)$ [8, 13]. Для последовательности σ, состоящей из конечного числа (s, d, t_s)-запросов, алгоритму Дейкстры потребуется $\mathcal{O}\left(|\sigma| \cdot |V|^2\right)$ времени. Это сопоставимо с оценкой времени работы модифицированного алгоритма ALT, приведенной в теореме 1.2. Заметим, что указанные оценки получены для худшего случая. Однако на практике время выполнения данных алгоритмов во многом зависит от содержания запросов (положения стартовых и целевых вершин относительно друг друга) и порядка их следования в σ. Время выполнения модифицированного алгоритма ALT также зависит от значения K и Δ. Поэтому экспериментально сравнивать эти алгоритмы между собой целесообразно на различных сетях и последовательностях σ, реально существующих или сгенерированных случайным образом. Для сравнения могут быть использованы следующие параметры: $|V_\sigma|$ — мощность пространства поиска решения для последовательности σ в целом; $|V_{avg}|$ — среднее значение мощности пространство поиска, вычисленное для одного запроса последовательности σ. Для данных параметров верны соотношения: $|V_{avg}| \leq |V_\sigma| \leq |V|$. Эти параметры отражают, на сколько тот или иной алгоритм охватывает исходную сеть в процессе своей работы.

Глава 2. Алгоритм RevTree приближенного решения задачи о кратчайшем пути в ресурсоограниченной сети

Большинство современных телекоммуникационных сетей являются мультисервисными, поскольку предоставляют пользователям множество разнообразных услуг, касающихся проводной телефонии, сотовой связи, кабельного телевидения и передачи данных. Качество обслуживания (Quality of Service, QoS) в мультисервисных сетях определяется такими параметрами как полоса пропускания, задержка, вариация задержки, стоимость и надежность передачи данных [78, 86]. Всякий запрос пользователя на оказание услуги предполагает определение некоторого (желательно оптимального по затратам, например, по времени или стоимости) маршрута между двумя узлами с учетом ресурсных возможностей этой сети. Помимо телекоммуникационных сетей, подобные ограничения могут возникать при маршрутизации в дорожных сетях. Основным весом является время, затрачиваемое на передвижение, в то время как дополнительными параметрами могут являться денежные средства, топливо или ограничения по скорости. Эти задачи можно сформулировать математически как задачу поиска ресурсоограниченного кратчайшего пути в графе, которая известна в литературе под названием Resource Constrained Shortest Path или кратко RCSP [50, 66].

Задача поиска кратчайшего пути в ресурсоограниченной сети предполагает определение оптимального маршрута в условиях весовой функции и дополнительных, заранее заданных, ресурсных ограничений. Известно, что задача поиска кратчайшего пути даже с одним ресурсным ограничением принадлежит к NP-трудным задачам [9, 50, 66].

Данная глава посвящена задаче поиска кратчайшего пути в ресурсоограниченной сети. Для решения данной задачи в сети с одним ресурсным ограничением предлагается алгоритм приближенного решения RevTree. Это соответствует задаче 2 диссертационного исследования. Основные результаты опубликованы в работах [5, 6, 24, 26, 30, 31, 34, 83, 85]. В параграфе 2.1 приведены определения необходимые для постановки задачи. В параграфе 2.2 приведен обзор известных подходов к решению задачи о кратчайшем пути

в ресурсоограниченной сети. В параграфе 2.3 предложен алгоритм RevTree для нахождения приближенного решения задачи о кратчайшем пути в ресурсоограниченной сети с одним ресурсом. Алгоритм отличается от ранее известных тем, что точность приближенного решения можно оценить исходя из параметров исходной сети. Для алгоритма сформулированы и доказаны теоремы о корректности и сложности.

2.1 Постановка задачи о кратчайшем пути в ресурсоограниченной сети

Рассмотрим задачу RCSP в теоретико-графовой постановке, используя общепринятые в теории графов терминологию и обозначения [10]. Пусть мультисервисная сеть описана взвешенным ориентированным графом (далее просто графом) $G = (V, E)$ без кратных дуг и петель, в котором каждая вершина $v \in V$ представляет узел сети, а каждая дуга $e \in E$ — канал связи между соответствующими узлами сети, при этом $n = |V|$ и $m = |E|$. Считаем, что на множестве дуг графа $G = (V, E)$ задана функция $w(e)\colon E \to \mathbb{R}^+$, ставящая в соответствие каждой дуге $e \in E$ ее вес $w(e) > 0$. Пусть для вершин $s, d \in V$ в графе $G = (V, E)$ существует путь P, идущий от вершины s к вершине d. Полагаем, что вес этого (s, d)-пути P вычисляется как сумма весов всех входящих в него дуг:

$$w(P) = \sum_{e \in P} w(e), \tag{2.1}$$

т.е. функция $w(e)$ является аддитивной. Если $w(e)$ — стоимость передвижения по дуге e, то $w(P)$ можно интерпретировать как стоимость прохождения (s, d)-пути P в графе $G = (V, E)$.

Пусть для каждой дуги графа G также заданы функции $r_i(e)\colon E \to \mathbb{R}^+$, $i = 1, \ldots, k$, отражающие ресурсные потребности, которые необходимы для передвижения по этой дуге, и всегда $r_i(e) > 0$. Предполагается, что все эти функции аддитивные, т. е. для любого (s, d)-пути P верны равенства:

$$r_i(P) = \sum_{e \in P} r_i(e), \quad i = 1, \ldots, k. \tag{2.2}$$

Кроме того, заданы величины $R_i \in \mathbb{R}^+, i = 1, \ldots, k$, определяющие ресурсные

ограничения рассматриваемой мультисервисной сети.

Всякий (s, d)-путь P называется допустимым, если он удовлетворяет ресурсным ограничениям:

$$r_i(P) \leq R_i, \quad i = 1, \ldots, k. \tag{2.3}$$

Оптимальным называется (s, d)-путь P, который минимизирует величину $w(P)$ в (2.1) и удовлетворяет ограничениям (2.3). Данные понятия проиллюстрированы на рис. 2.1.

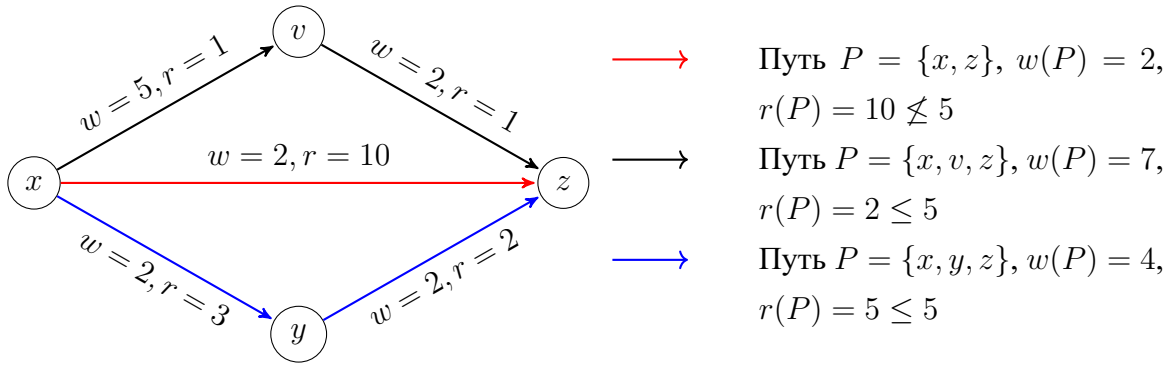


Рисунок 2.1 — В сети с ресурсным ограничением $R = 5$ красный путь от вершины x до вершины z не является допустимым, черный является допустимым, а синий является оптимальным

Задача поиска кратчайшего пути в ресурсоограниченной сети (Resource Constrained Shortest Path problem, RCSP) имеет следующий вид.

Задан: граф $G = (V, E)$, на дугах которого определены положительные вещественнозначные функции $w(e)$, $r_i(e)$, величины R_i, $i = 1, \ldots, k$.

Требуется: найти в $G = (V, E)$ оптимальный (s, d)-маршрут.

Всякий допустимый (s, d)-путь можно рассматривать в качестве приближенного решения данной задачи. Очевидно, что если множество допустимых (s, d)-путей пустое, то задача RSCP не имеет решения. Такая ситуация возникает, когда заданы жесткие ресурсные ограничения, например, когда $\min\limits_{e \in E} r_i(e) > R_i$. Для достаточно больших значениях ресурсных ограничений, например, при $\sum\limits_{e \in E} r_i(e) \leq R_i$, задача RCSP вырождается в классическую задачу о кратчайшем пути. Важно отметить, что для фиксированного числа

ресурсов и неотрицательности всех функций $w(e), r_i(e)$ задача RCSP является NP-трудной в слабом смысле [66, 75]. Если рассматривается только один ресурс с $r(e) = 1$ для всех дуг графа, то задача RCSP сводится к поиску кратчайшего (s, d)-пути, содержащего не более R дуг. В этом случае она разрешима за полиномиальное время [9]. Для задачи с ограниченным ресурсом кратчайший путь в большинстве случаев не является оптимальным.

2.2 Обзор подходов к решению задачи RCSP

Задача поиска кратчайшего пути в ресурсоограниченной сети, является расширением классической задачи поиска кратчайшего пути в графе. Постановка задачи о кратчайшем пути в ресурсоограниченной сети допустима в терминах целочисленного линейного программирования [63]. Таким образом, в настоящее время выделяют три класса методов и соответствующих им алгоритмов, способных находить точное или приближенное решение задачи RCSP (табл. 2.1): методы ранжирования путей [53], методы маркировки вершин [51,61,66,86,91], методы лагранжевой релаксации [63,65,75]. Первые два класса методов основаны на теоретико-графовой постановке задачи, в то время как методы третьего класса исходят из постановки задачи RCSP на языке целочисленного линейного программирования [63]. Большинство алгоритмов, основанных на теоретико-графовой постановке задачи, являются расширениями алгоритма Дейкстры или подобных ему [51, 63, 86].

Таблица 2.1 — Методы решения задачи RCSP

Метод	Представитель	Сложность
Ранжирование путей	Алгоритм Йена [53]	$\mathcal{O}\left(k \cdot n \cdot (m + n \cdot \log n)\right)$
Маркировка вершин	Алгоритм $P + RT$ [40]	$\mathcal{O}\left(m \cdot n \cdot \left(\frac{1}{\varepsilon} + \log n\right)\right)$
	$(1 + \varepsilon)(K - 1)$-приближенный алгоритм [88]	$\mathcal{O}\left(n \cdot m \cdot \log\log\log n + m \cdot \left(\frac{n}{\varepsilon}\right)^{k-1}\right)$

Методы ранжирования путей предполагают решения задачи k-кратчайших путей в графе. В задаче требуется найти не только кратчайший путь, но и последующие $k - 1$ кратчайших путей, которые могут отличаться от крат-

чайшего и быть тяжелее в смысле весовой функции. Поскольку оптимальный путь для задачи RCSP может отличаться от кратчайшего пути, в смысле ресурсной функции, то параметр k изначально не может быть определен. Таким образом, подход поиска k-кратчайших путей усложняется до перебора всех кратчайших путей до тех пор, пока не найдется допустимый. Наиболее популярный алгоритм в этом классе методов — алгоритм Йена, который находит решение задачи k-кратчайших путей за время $\mathcal{O}\big(k \cdot n \cdot (m + n \cdot \log n)\big)$, где k может экспоненциально зависеть от размера исходной сети [53].

Методы маркировки вершин предполагают использование дополнительных меток для корректной маршрутизации в ресурсоограниченной сети. В этом классе методов присутствуют как точные алгоритмы решения, так и приближенные. Алгоритм точного решения обладает сложностью $\mathcal{O}\left(n^5 \cdot b \log(n \cdot b)\right)$, где b — наибольший вес ребра в графе [86]. Время выполнения данного алгоритма велико, поэтому востребованы исследования приближенных алгоритмов решения задачи. Приближенные алгоритмы разделяют на три вида: приближенные по весовой функции, приближенные по ресурсным функциям, приближенные по комбинации весовой и ресурсной функций. Алгоритм, представленный в работе [88], обладает следующей сложностью $\mathcal{O}\left(n \cdot m \cdot \log\log\log n + m \cdot \left(\frac{n}{\varepsilon}\right)^{k-1}\right)$, где k — число ресурсных ограничений. Он находит кратчайший путь, который отличается от оптимального не более чем в $(1 + \varepsilon)$ раз. Алгоритмы, приближенные по ресурсным функциям, имеют широкое применение в телекоммуникационных сетях. Один из эффективных алгоритмов приближенного решения предложен в работе [40] и имеет вычислительную сложность $\mathcal{O}\left(m \cdot n \cdot \left(\frac{1}{\varepsilon} + \log n\right)\right)$. Алгоритм находит путь отличный от оптимального не более чем в $\Big(3 \cdot (1 + \varepsilon), 2\Big)$ раз.

Методы лагранжевой релаксации в основном состоят из трех фаз. На первой фазе вычисляются нижние и верхние границы оптимальных значений для задачи. На второй фазе эти границы используются для упрощения исходного графа. На третьей фазе закрывается разрыв между границами путем нахождения оптимального пути некоторым алгоритмом поиска кратчайшего пути. Алгоритмы этого направления представлены в работах [47, 63, 65, 75].

2.3 Приближенный алгоритм RevTree

Здесь и далее, под размерностью задачи RCSP понимаются значения n и m — число вершин и дуг графа $G = (V, E)$ соответственно, а под её параметрами — значения функций $w(e), r_i(e)$. Для краткости вместо $r_i(e)$ и R_i будем писать $r(e)$ и R. Опишем алгоритм RevTree для случая, когда имеется только один ресурс R_i и $w(e), r_i(e)$ — положительные вещественнозначные функции.

В работе предлагается модернизированный алгоритм Дейкстры с потенциальными функциями RevTree. Алгоритм Дейкстры использует понятие окрестности вершины графа $G = (V, E)$, которое трактуется следующим образом: окрестность вершины $v \in V$ — множество $\Gamma(v)$ концов дуг, исходящих из вершины v, и обладающих временными метками. Множество $\Gamma(v)$ определяет возможные направления дальнейшего движения по дугам графа $G = (V, E)$ из вершины v. Алгоритм RevTree уменьшает мощность множества $\Gamma(v)$ с учетом ресурсного ограничения R, что позволяет находить решение задачи RCSP. Схема алгоритма RevTree представлена на рис. 2.2.

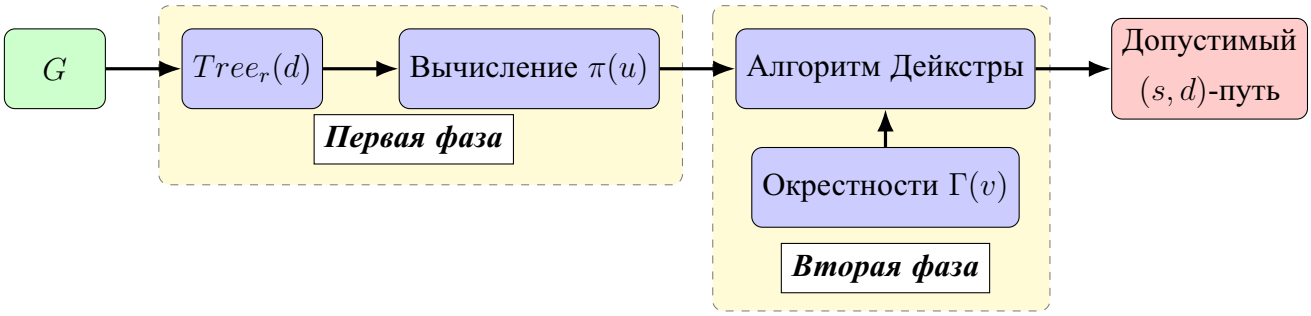


Рисунок 2.2 — Схема алгоритма RevTree

Алгоритм RevTree состоит из двух фаз, на каждой из которых однократно выполняется алгоритм Дейкстры. На первой фазе находится дерево путей минимального веса, вычисляемых по весовой функции $r(e), e \in E$. Корнем данного дерева является целевая вершина d. Такое дерево определяет для каждой вершины $u \in V$, в том числе из окрестности $\Gamma(v)$, (u, d)-путь, вес которого определяет минимальный ресурс, необходимый для прохождения этого пути. Обозначим этот путь через P_2. Пусть P_1 — путь из стартовой вершины s в текущую вершину v, найденный как некоторое решение задачи RCSP для вершин

s и v. Согласно формуле (2.2) для прохождения этого (s, v)-пути был затрачен ресурс величины $r(P_1)$. Тогда для перемещения из вершины v в вершину $u \in \Gamma(v)$ необходимо выполнение условия:

$$r(P_1) + r(v, u) + \pi(u) \leq R, \tag{2.4}$$

где $\pi(u) = \sum\limits_{e \in P_2} r(e)$. Условие (2.4) гарантирует, что путь (P_1, e, P_2), где $e = (v, u) \in E$, является допустимым решением задачи RCSP. Условие (2.4) проиллюстрировано на рис. 2.3.

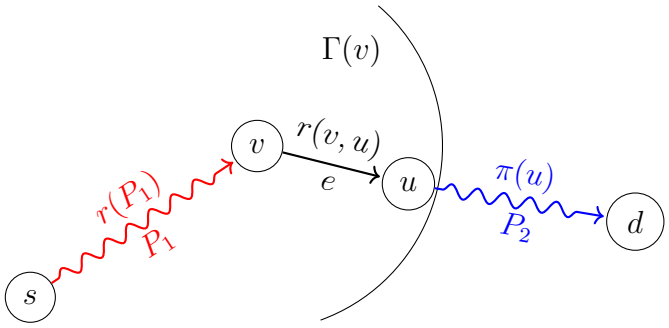


Рисунок 2.3 — Путь (P_1, e, P_2) допустимый, если ресурсные потребности красного пути $r(P_1)$, дуги $r(e)$ и потенциального ресурса синего пути $\pi(u)$ удовлетворяет неравенству (2.4)

На второй фазе выполняется алгоритм Дейкстры с усеченными окрестностями вершин. Усечение окрестности $\Gamma(v)$ для каждой текущей вершины $v \in V$ выполняется по условию (2.4): если для вершины $u \in \Gamma(v)$ не выполняется условие (2.4), то она удаляется из $\Gamma(v)$.

Описание алгоритма RevTree на псевдокоде представлено в алгоритме 2.1. По построению алгоритм RevTree всегда находит допустимое решение задачи RCSP, если оно существует для заданных вершин s и d.

Теорема 2.1 (О сложности алгоритма RevTree)**.** *Алгоритм* RevTree *требует* $\mathcal{O}\left(|V|^2\right)$ *времени и* $\mathcal{O}\left(|V|\right)$ *памяти для решения задачи RCSP с одним ресурсным ограничением.*

Доказательство. Известно, что алгоритм Дейкстры требует $\mathcal{O}\left(|V|^2\right)$ времени и $\mathcal{O}(|V|)$ памяти [13, 21]. На первой фазе с помощью алгоритма Дейкстры находится дерево путей минимального веса с корнем в вершине d, для сохранения которого необходимо $\mathcal{O}(|V|)$ памяти. На второй фазе в алгоритм Дейкстры

Алгоритм 2.1. Алгоритм RevTree

Вход: граф $G = (V, E)$, функции $w(e)$ и $w_r(e)$, величина W_r, стартовая вершина s, целевая вершина d

Выход: значение веса и последовательность вершин от s до d образующих допустимый путь

Инициализация

$w[s] \leftarrow 0$

$r[s] \leftarrow 0$

$p[s] \leftarrow 0$

$Passed \leftarrow \oslash$

$\pi_r[d] \leftarrow 0$

Первая фаза

Выполнить для всех $v \in V \neq d$ $\pi_r[v] \leftarrow \infty$

Конец цикла

До тех пока $\exists u \notin Passed$

    Пусть $u \notin Passed$ с минимальным $\pi_r[u]$

    $Passed \leftarrow u$

    Выполнить для всех $v \notin Passed$ & $e = (u, v) \in E$

        Если $\pi_r[v] > \pi_r[u] + r(e)$ **то**

            $\pi_r[v] \leftarrow \pi_r[u] + r(e)$

        Конец условия

    Конец цикла

Конец цикла

$Passed \leftarrow \oslash$

Вторая фаза

Выполнить для всех $v \in V \neq s$ $w[v] \leftarrow \infty$ $p[v] \leftarrow 0$ $r[v] \leftarrow 0$

Конец цикла

До тех пока $\exists u \notin Passed$

    Пусть $u \notin Passed$ с минимальным $w[u]$ и $r[u] + \pi_r[v] < R$

    $Passed \leftarrow u$

    Выполнить для всех $v \notin Passed$ & $e = (u, v) \in E$

        Если $w[v] > w[u] + w(e)$ **то**

            $w[v] \leftarrow w[u] + w(e)$

            $r[v] \leftarrow r[u] + r(e)$

            $p[v] \leftarrow u$

        Конец условия

    Конец цикла

Конец цикла

добавляется проверка условия (2.4), которая не влияет на оценку вычислительной сложности данного алгоритма. Поскольку обе фазы алгоритма RevTree выполняются последовательно, то в целом для нахождения решения задачи RCSP затрачивается $\mathcal{O}\left(|V|^2\right)$ времени и $\mathcal{O}\left(|V|\right)$ памяти. □

Получим оценку точности приближенного решения, формируемого алгоритмом RevTree. Для каждой дуги $e \in E$ определим функцию

$$\lambda(e) = \frac{r(e)}{w(e)}.$$

Вычислим минимальное и максимальное значение этой функции на множестве дуг E:

$$\lambda_{\min} = \min_{e \in E} \lambda(e) > 0,$$
$$\lambda_{\max} = \max_{e \in E} \lambda(e) > 0.$$

Заметим, что функция $\lambda(e)$ определена для любого $e \in E$, поскольку в формулировке задачи RCSP предполагается, что $w(e) > 0, r(e) > 0, e \in E$.

Теорема 2.2 (О точности решения алгоритма RevTree)**.** *Алгоритм* RevTree *находит ε-приближенное решение задачи RCSP с одним ресурсным ограничением, если оно существует для вершин s и d. Точность решения определяется как*

$$\varepsilon = \frac{\lambda_{\max}}{\lambda_{\min}} - 1,$$

где

$$\lambda_{\min} = \min_{e \in E} \left(\frac{r(e)}{w(e)} \right) > 0, \quad \lambda_{\max} = \max_{e \in E} \left(\frac{r(e)}{w(e)} \right) > 0.$$

Доказательство. В рамках доказательства обозначим вес некоторого оптимального пути как $\mathcal{OPT}$. Пусть алгоритм RevTree на некотором шаге нашёл путь P_1, идущий из стартовой вершины s в текущую вершину v, как оптимальное решение задачи RCSP для вершин s и v. Если вершина v совпадает с целевой вершиной d, то найдено оптимальное решение исходной задачи. Если вершина v совпадает с s, то путь P_1 содержит пустое множество дуг и для него $w(P_1) = r(P_1) = 0$. В общем случае вершина v определяет начало ещё не пройденной части искомого (s, d)-пути. Рассмотрим неусечённую окрестность

$\Gamma(v)$ текущей вершины v как множество концов дуг, исходящих из вершины v и обладающих временными метками. Пусть $x \in \Gamma(v)$ — вершина, для которой $w(v,x) = \min\limits_{u \in \Gamma(v)} w(v,u)$, но не выполняется условие (2.4), т. е. верно соотношение

$$r(P_1) + r(v,x) + \pi(x) > R. \tag{2.5}$$

Именно эту вершину выбирает алгоритм Дейкстры, если окрестность $\Gamma(v)$ не подверглась усечению. При выборе вершины $u \in \Gamma(v)$, удовлетворяющей условию (2.4), алгоритм RevTree отклоняется от кратчайшего пути в смысле (2.1) без учёта ресурсного ограничения (2.2). Обозначим (v,d)-путь, найденный с помощью неусечённых окрестностей, как P^*_{rest}, а с помощью усечённых окрестностей — P_{rest}. Для путей (P_1, P_{rest}) и (P_1, P^*_{rest}) справедливы следующие неравенства:

$$w(P_1) + w(P^*_{\text{rest}}) \le \mathcal{OPT} \le w(P_1) + w(P_{\text{rest}}). \tag{2.6}$$

В (2.6) величина $w(P_1) + w(P^*_{\text{rest}})$ определяет значение целевой функции для кратчайшего пути без учёта ресурсных ограничений, а $w(P_1) + w(P_{\text{rest}})$ — значение целевой функции пути, найденного алгоритмом RevTree. Предполагается, что решение задачи RCSP существует. Следовательно, найдётся хотя бы один допустимый (s,d)-путь, а значит, всегда существует определённый выше путь P_{rest}. Для прохождения всякого (v,d)-пути доступен ресурс $R_{\text{rest}} = R - r(P_1)$. Согласно условию (2.4), справедливо неравенство $R_{\text{rest}} \ge \pi(v)$. Исходя из определения функции $\lambda(e)$, для всякой дуги $e \in E$ имеют место соотношения

$$0 < \lambda_{\min} \cdot w(e) \le r(e) \le \lambda_{\max} \cdot w(e).$$

Поскольку функции $w(e)$ и $r(e)$ аддитивные, получим подобные соотношения для любого (v,d)-пути, в том числе P^*_{rest} и P_{rest}:

$$0 < \lambda_{\min} \cdot w(P^*_{\text{rest}}) \le r(P^*_{\text{rest}}) \le \lambda_{\max} \cdot w(P^*_{rest}); \tag{2.7}$$

$$0 < \lambda_{\min} \cdot w(P_{\text{rest}}) \le r(P_{\text{rest}}) \le \lambda_{\max} \cdot w(P_{\text{rest}}). \tag{2.8}$$

Путь P^*_{rest} по построению проходит через вершину x, поэтому с учётом (2.5)

верны соотношения

$$r(P^*_{\text{rest}}) \geq r(v,x) + \pi(x) > R - r(P_1) = R_{\text{rest}}.$$

Здесь $\pi(x)$ — минимально необходимый ресурс для передвижения из вершины x в вершину d. Следовательно, $r(P^*_{\text{rest}}) > R_{\text{rest}}$. Отсюда с учётом (2.7) получим

$$\frac{R_{\text{rest}}}{\lambda_{\max}} < w(P^*_{\text{rest}}). \tag{2.9}$$

Из соотношений (2.8) имеем неравенство $w(P_{\text{rest}}) \leq r(P_{\text{rest}})/\lambda_{\min}$. Поскольку всегда $w(P^*_{\text{rest}}) \leq w(P_{\text{rest}})$ и $r(P_{\text{rest}}) \leq R_{\text{rest}}$, то

$$w(P^*_{\text{rest}}) \leq \frac{r(P_{\text{rest}})}{\lambda_{\min}} \leq \frac{R_{\text{rest}}}{\lambda_{\min}}. \tag{2.10}$$

Из (2.10) и (2.10) следует

$$\frac{R_{\text{rest}}}{\lambda_{\max}} < w(P^*_{\text{rest}}) \leq \frac{R_{\text{rest}}}{\lambda_{\min}}. \tag{2.11}$$

Оценим решение, найденное алгоритмом RevTree, исходя из неравенств (2.6). В результате получим

$$\frac{w(P_1) + w(P_{\text{rest}}) - \mathcal{OPT}}{\mathcal{OPT}} \leq \frac{w(P_{\text{rest}}) - w(P^*_{\text{rest}})}{w(P_1) + w(P^*_{\text{rest}})}. \tag{2.12}$$

Наибольшее отклонение найденного решения от оптимального достигается при $v = s$ и $w(P_{\text{rest}}) = R_{\text{rest}}/\lambda_{\min}$. Тогда неравенство (2.12) принимает вид

$$\frac{\dfrac{R_{\text{rest}}}{\lambda_{\min}} - \mathcal{OPT}}{\mathcal{OPT}} \leq \frac{\dfrac{R_{\text{rest}}}{\lambda_{\min}} - w(P^*_{\text{rest}})}{w(P^*_{\text{rest}})}.$$

Согласно (2.11), справедливо $R_{\text{rest}}/\lambda_{\max} < w(P^*_{\text{rest}})$. Отсюда окончательно имеем

$$\frac{\dfrac{R_{\text{rest}}}{\lambda_{\min}} - \mathcal{OPT}}{\mathcal{OPT}} \leq \frac{\dfrac{R_{\text{rest}}}{\lambda_{\min}} - \dfrac{R_{\text{rest}}}{\lambda_{\max}}}{\dfrac{R_{\text{rest}}}{\lambda_{\max}}} = \frac{\lambda_{\max}}{\lambda_{\min}} - 1 = \varepsilon.$$

Таким образом, допустимое решение, найденное алгоритмом RevTree, отклоняется от оптимального решения задачи RCSP не более чем на величину

$$\varepsilon = \frac{\lambda_{\max}}{\lambda_{\min}} - 1.$$

□

Полученная в теореме 2.1 теоретическая оценка сложности алгоритма RevTree сопоставима с известными алгоритмами [51, 61, 66, 86, 91].

2.4 Выводы по главе 2

Основным результатом главы является алгоритм RevTree для приближенного решения задачи о кратчайшем пути в ресурсоограниченной сети с одним ресурсом.

Следует отметить следующие особенности алгоритма RevTree.

1. Точность алгоритма зависит от значений $\lambda_{\min}$ и $\lambda_{\max}$, которые вычисляются на основе весовых и ресурсных функций дуг. Приведена теоретическая оценка алгоритма RevTree в теореме 2.1. Точность алгоритма RevTree представлена теоремой 2.2. Полученная оценка для алгоритма RevTree, с учетом его точности, сопоставима со сложностью задачи RCSP с одним ресурсом.

2. Заметим, что исходя из способа определения значений $\lambda_{\min}$ и $\lambda_{\max}$ в теореме 2.2 допустимо повышение точности алгоритма RevTree путем применения приема масштабирования дуг сети. Прием заключается в том, что если весовые или ресурсные функции дуг допускают масштабирование, т. е. уменьшение или увеличение соответствующих весов, то можно подобрать такой масштаб, который будет максимизировать значение ε. Метрические сети допускают масштабирование, например, дорожные сети, но следует учесть, что такой прием может исказить результат решения задачи.

Глава 3. Задачи перечислительного типа на гиперграфах

Задачи перечислительного типа для сетей, представленных гиперграфом, предполагают нахождение всех конфигураций гиперграфа с заданными условиями. Под конфигурацией понимается любая система подмножеств конечного множества [35]. В главе рассматриваются задачи поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа и поиска всех максимальных индуцированных биклик гиперграфа. Первая задача принадлежит классу $\#P$-полных задач [16, 71, 72], поскольку решение задачи экспоненциально зависит от размера входного графа. Вторая задача не легче, чем NP-трудная задача, поскольку задача поиска даже одной максимальной биклики принадлежит к классу NP-трудных задач [38, 74].

Данная глава соответствует задаче 3 диссертационного исследования. Основные результаты опубликованы в работах [22, 28, 84]. В параграфе 3.1 представлены основные понятия и обозначения теории гиперграфов и $(0,1)$-матриц. Приведены постановки задач о перечислении максимальных индуцированных биклик гиперграфа и поиске всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа. Доказана теорема о связи двудольного подгиперграфа и двудольного подграфа соответствующего вершинного графа. В параграфе 3.2 выполнен обзор известных подходов к решению поставленных задач перечислительного типа. В параграфе 3.3 предложен алгоритм HFINDMCS для решения задачи поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа. Алгоритм основан на гиперграфовом подходе и последовательной генерации полных подматриц. Для алгоритма HFINDMCS доказана теорема о теоретической сложности. В параграфе 3.4 предложен алгоритм HFINDMIB генерации всех максимальных индуцированных биклик гиперграфа. Алгоритм основан на гиперграфовом подходе, специальном виде матрицы смежности двудольного подграфа и теореме, доказанной в параграфе 3.1. Алгоритм HFINDMIB наследует идеи алгоритма HFINDMCS и использует в своем процессе генерацию допустимых индуцированных биклик, из которых выбираются максимальные. Доказана теорема о теоретической сложности алгоритма HFINDMIB.

3.1 Постановка задач перечислительного типа на гиперграфах

Сформулируем задачи поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа и поиска всех максимальных индуцированных биклик гиперграфа, используя общепринятые в теории гиперграфов терминологию и обозначения [1, 11, 17, 18, 35].

Задан (n,m)-гиперграф $H = (X,U)$ и $(0,1)$-матрица инцидентности I, где X — конечное множество вершин и $|X| = n$, U — конечное семейство гиперребер гиперграфа и $|U| = m$. Множество $X(u)$ содержит все вершины, инцидентные гиперребру $u \in U$. Множество $U(x)$ содержит все гиперребра, инцидентные вершине $x \in X$. Одним из способов задания гиперграфа является $(0,1)$-матрица инцидентности I, где 1 ставится в случае, когда гиперребро содержит вершину, и 0 в противном случае. Будем называть степенью гиперребра $u \in U$ мощность множества $|X(u)|$. Полной подматрицей $(0,1)$-матрицы называется подматрица, все без исключения элементы которой равны 1. Максимально полной подматрицей называется полная подматрица, не содержащая никакой другой максимально полной подматрицы.

Определение 3.1. Подгиперграфом, индуцированным множеством вершин X', называется $H' = (X', U')$, где $U' = \{u' \colon X(u') = X(u) \cap X' \neq \varnothing, u \in U\}$.

Известно определение двудольности гиперграфа, которое аналогично 2-раскраске [92]: гиперграф $H = (X,U)$ называется двудольным, когда множество вершин X может быть разделено на два множества S_0 и S_1 таким образом, что $S_0 \cup S_1 = X$, $S_0 \cap S_1 = \varnothing$, и для всякого гиперребра $u \in U$ выполнено $|X(u) \cap S_0| = 1$. В работе вводится ослабленное определение двудольности.

Определение 3.2. Подгиперграф $H' = (X', U')$, индуцированный множеством вершин X', является двудольным, если существует разбиение $S_0 \cup S_1 = X'$ и $S_0 \cap S_1 = \varnothing$, такое, что для любого гиперребра $u' \in U'$ выполнено $|S_0 \cap X(u')| \leq 1$ и $|S_1 \cap X(u')| \leq 1$.

Определение 3.3. Вершинным графом гиперграфа $H = (X,U)$ называется граф $L_2(H) = (X,E)$, множество вершин которого совпадает с множеством

вершин X гиперграфа H, при этом две вершины графа $L_2(H)$ смежны тогда и только тогда, когда смежны соответствующие им вершины гиперграфа H.

На рис. 3.1 представлены гиперграф H и соответствующий ему вершинный граф $L_2(H)$.

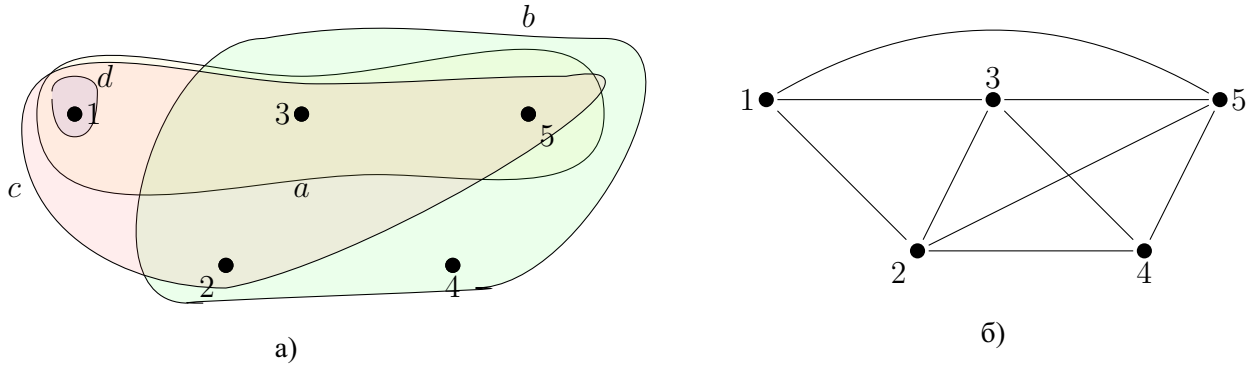


Рисунок 3.1 — а) гиперграф $H = (X, U)$, где $X = \{1, 2, 3, 4, 5\}$ и $U = \{a, b, c, d\}$; б) вершинный граф $L_2(H) = (X, E)$

Сформулируем и докажем теорему об эквивалентности индуцированных двудольных подгиперграфов гиперграфа H и двудольных подграфов вершинного графа $L_2(H)$.

Теорема 3.1 (Об эквивалентности индуцированных двудольных подгиперграфов гиперграфа H и двудольных подграфов вершинного графа $L_2(H)$)**.** *Подгиперграф $H' = (X', U')$ является двудольным тогда и только тогда, когда в вершинном графе $L_2(H)$ гиперграфа H существует двудольный подграф, индуцированный множеством вершин X'.*

Доказательство. Докажем достаточность: если в гиперграфе H найдется двудольный подгипреграф $H' = (X', U')$, то в $L_2(H)$ существует двудольный подграф, индуцированный множеством X'.

Доказательство следует непосредственно из определений 3.1 – 3.3. Гиперребро $u \in U$ исходного гиперграфа $H = (X, U)$ порождает гиперребро u' в гиперграфе H'. Тогда согласно определению 3.1 гиперребро u может обладать большей степенью чем u', т.е. $|u| \geq |u'|$. Однако, из определения 3.2 следует, что для двудольного подгиперграфа $H' = (X', U')$ степень всякого ребра $u' \in U'$ не превосходит двух. Это обусловлено тем, что если

$|X(u')| > 2$, то нарушается требование $|S_0 \cap X(u')| \leq 1$, $|S_1 \cap X(u')| \leq 1$, где $S_0 \cup S_1 = X'$ и $S_0 \cap S_1 = \oslash$. Таким образом, если рассматриваемый подгиперграф $H' = (X', U')$ является двудольным, то согласно определению 3.3 его вершинная реализация $L_2(H')$ — двудольный граф. С другой стороны, вершинная реализация $L_2(H')$ является подграфом $L_2(H)$, индуцированным множеством X'. Следовательно, $L_2(H')$ есть искомый двудольный подграф вершинной реализации $L_2(H)$ исходного гиперграфа.

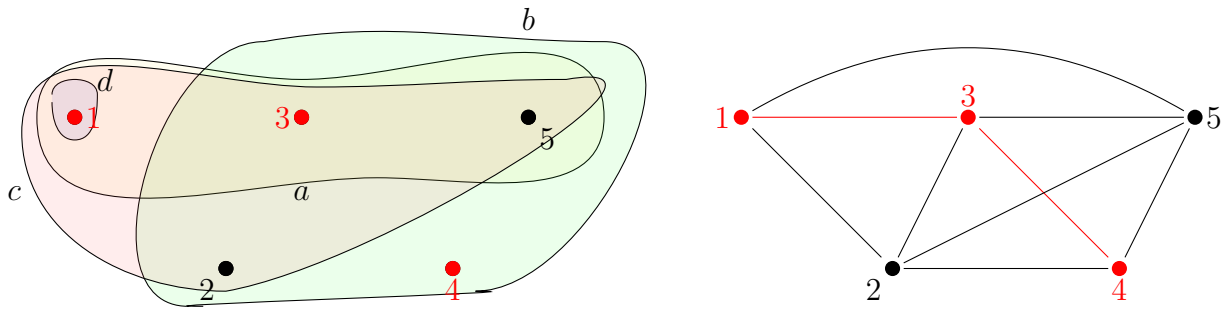
Докажем необходимость: если в вершинной реализации $L_2(H)$ исходного гиперграфа H существует двудольный подграф, индуцированный множеством X', то найдется двудольный подгипреграф $H' = (X', U')$.

Пусть в графе $L_2(H)$ существует двудольный подграф на множестве вершин X' с долями S_0 и S_1, где $S_0 \cup S_1 = X'$ и $S_0 \cap S_1 = \oslash$. В гиперграфе $H = (X, U)$ рассмотрим подгиперграф $H' = (X', U')$, индуцированный множеством X'. Согласно определению 3.1 в подгиперграфе H' множество гиперребер имеет вид $U' = \{u' \colon X(u') = X(u) \cap X' \neq \oslash, u \in U\}$. Из определения 3.3 следует, что для любого гиперребра $u \in U$ множество $X(u)$ образует полный подграф в вершинном графе $L_2(H)$. Известно, что любой двудольный граф не имеет полных подграфов, содержащих более двух вершин [10, 61]. Отсюда всякое гиперребро $u' \in U'$ будет удовлетворять неравенствам $|X(u') \cap S_0| \leq 1$ и $|X(u') \cap S_1| \leq 1$ и $|u'| \leq 2$. □

Положение теоремы 3.1 проиллюстрировано на рис. 3.2.

Двудольный граф называется полным двудольным графом (бикликой), если всякая вершина одной доли соединена со всеми вершинами второй доли [10]. Данное определение можно сформулировать иначе, если двудольный граф содержит все возможные ребра, не нарушающие условие двудольности, то такой граф называется полным двудольным графом. К поиску биклик сводится ряд теоретико-графовых задач [9, 16], которые относятся к классу $\sharp P$-полных или NP-полных задач, поскольку в общем случае число максимальных биклик экспоненциально зависит от размера графа [72, 79].

Далее под индуцированной бикликой гиперграфа будем понимать полный двудольный индуцированный подгиперграф исходного гиперграфа H.



а) б)

Рисунок 3.2 — а) двудольный подгиперграф H' гиперграфа $H = (X, U)$ с долями $S_0 = \{1, 4\}$ и $S_1 = \{3\}$ выделен красным цветом; б) вершинный двудольный подграф $L_2(H')$ с долями $S_0 = \{1, 4\}$ и $S_1 = \{3\}$ выделен красным цветом

Определение 3.4. Подгиперграф $H' = (X', U')$, индуцированный множеством вершин X', такой, что $U' = \{u\colon s_0, s_1 \in X(u), s_0 \in S_0, s_1 \in S_1\}$ и $S_0 \cup S_1 = X'$, $S_0 \cap S_1 = \varnothing$ называется полным двудольным индуцированным подгиперграфом исходного гиперграфа H.

Максимальная биклика — это биклика, которая не может быть расширена путем включения дополнительных смежных вершин, или, что эквивалентно, для максимальной биклики не существует другой биклики, которая полностью включает данную. Это справедливо как для графов, так и для гиперграфов [92].

Задача поиска всех максимальных индуцированных биклик гиперграфа (Maximal Induced Bicliques Generation Problem for Hypergraphs, MIBGP for Hypergraphs) имеет следующий вид.

Задан: гиперграф $H = (X, U)$ без кратных гиперребер.
Требуется: найти множество всех максимальных индуцированных биклик.

Заметим, что задача нахождения максимальных индуцированных биклик в графе тесно связана с поиском подматриц специального вида [39]. Продемонстрируем связь между задачей нахождения максимальных индуцированных биклик в гиперграфе и задачей поиска максимально полных подматриц $(0, 1)$-матрицы.

Для двудольного подгиперграфа $H' = (X', U')$ запишем его эквивалентное представление в виде $(0,1)$-матрицы смежности вершинного графа $L_2(H')$. Обозначим S_0, S_1 — доли подгиперграфа H', мощности которых равны c и d, соответственно. Поскольку вершинный граф $L_2(H')$ тоже является двудольным, его матрица смежности представима в виде [39]:

$$A' = \begin{pmatrix} O_c & B' \\ B'^T & O_d \end{pmatrix}, \tag{3.1}$$

где O_c, O_d — нулевые матрицы порядка c и d, соответственно, а B' матрица размера $c \times d$, которая отражает смежность вершин между долями S_0 и S_1. Очевидно, что в случае, когда подгиперграф H' является бикликой, то матрица B является полной подматрицей матрицы A. Известно, что проверка всякой максимально полной подматрицы на соответствие виду (3.1) выполнима за линейное время относительно размера подматрицы [13, 39]. Таким образом, для поиска индуцированных биклик в гиперграфе $H = (X, U)$ требуется найти такие полные подматрицы B' матрицы смежности A вершинного графа $L_2(H)$, для которых будет существовать подматрица вида (3.1), что приводит к задаче поиска всех максимально полных подматриц (maximally complete submatrices) исходной $(0,1)$-матрицы. Задача поиска всех максимально полных подматиц $(0,1)$-матрицы инцидентности гиперграфа (Maximally Complete Submatrices Problem for Hypergraph, MCSP for Hypergraph) имеет следующую постановку.

Задан: гиперграф $H = (X, U)$ без кратных гиперребер с $(0,1)$-матрицей инцидентности I.

Требуется: найти множество всех максимально полных подматриц $(0,1)$-матрицы инцидентности I.

Заметим, что максимально полные подматрицы могут представлять различные комбинаторные объекты [35], а задача их поиска носит перечислительный характер и принадлежит классу сложности $\sharp P$-полных задач [9, 16, 35, 55, 71, 72]. Применение алгоритмов решения задачи выделения максимально полных подматриц ограничивается высокой трудоемкостью на разреженных матрицах большой размерности.

3.2 Обзор подходов к решению перечислительных задач на графах и гиперграфах

В работе исследуются перечислительные задачи на графах и гиперграфах. При этом стоит отметить, что значительная часть алгоритмов для нахождения всех максимальных биклик предложена именно для графов. Актуальные алгоритмы решения этих задач приведены в табл. 3.1.

Таблица 3.1 — Алгоритмы решения перечислительных задач

Задача	Представитель	Сложность
MCSP	Close-by-One [71]	$\mathcal{O}\left(\lvert\mathcal{MCS}\rvert \cdot m \cdot n^2\right)$
MIBGP	Алгоритм LCM-MBC [73]	$\mathcal{O}\left(n \cdot m \cdot \mathcal{N}\right)$
MIBGP	Алгоритм Хермелина [62]	$\mathcal{O}\left(n \cdot k \cdot (\Delta + k) \cdot 3^{\frac{\Delta+k}{3}}\right)$

Задача поиска всех максимально полных подматриц соответствует другой хорошо известной задаче, а именно задаче поиска всех формальных понятий формального контекста. Поиск всех формальных понятий хорошо изучен в работах [3, 16, 55, 71, 76]. Формальное понятие является максимально полной подматрицей $(0, 1)$-матрицы формального контекста [55]. Для этой задачи разработано множество алгоритмов, но наиболее востребованный из них Close-by-One (CbO) [71]. Асимптотическая сложность данного алгоритма оценивается как $\mathcal{O}\left(|\mathcal{MCS}| \cdot m \cdot n^2\right)$, где $\mathcal{MCS}$ — множество всех максимально полных подматриц.

Задача поиска всех максимальных биклик активно изучается для сетей представленных графами или гиперграфами. Выделяют два направления исследований, поиск всех максимальных неиндуцированных биклик и максимальных индуцированных биклик. Индуцированная биклика $X' = S_0 \cup S_1$ предполагает, что доли S_0 и S_1 являются независимыми множествами, т. е. каждое множество является внутренне несмежным. Если условие независимости множества отбрасывается, то такие биклики называются неидуцированными. Известно, что для графов с произвольной древестностью, сложность по времени нахождения всех максимальных индуцированных биклик не превосходит $\mathcal{O}\left(a^3 \cdot 2^{2a} \cdot n\right)$, где a — число древесности графа [52]. Ряд исследовате-

лей отмечают, что на практике востребованы только максимальные неиндуцированные биклики большой размерности [44,73,90]. В работе [73] был предложен алгоритм для поиска максимальных неиндуцированных биклик, в котором можно установить порог размерности p биклик. Сложность данного алгоритма составляет $\mathcal{O}(n \cdot m \cdot \mathcal{N})$, где $\mathcal{N}$ — множество всех неиндуцированных биклик размерности большей или равной p. Для поиска всех максимальных индуцированных биклик графа в работе [62] предложен алгоритм со сложностью $\mathcal{O}\left(n \cdot k \cdot (\Delta + k) \cdot 3^{\frac{\Delta+k}{3}}\right)$, где Δ — максимальная степень вершины графа, а k — вырожденность графа.

Задача поиска всех максимальных биклик для графов может быть расширена на гиперграфы. В частности, для бигиперграфов задача поиска всех максимальных индуцированных биклик исследована в [92]. Под бигиперграфом понимается гиперграф вида $H = \left(H^0, H^1\right)$, где всякое гиперребро гиперграфов H^0 и H^1 содержится в H. При этом вводится определение двудольного гиперграфа согласно устойчивости множества вершин. Множество вершин $S \subseteq X$ устойчиво в H^i, если S не содержит гиперребер из H^i, $i = 0, 1$. Гиперграф $H = \left(H^0, H^1\right)$ называется двудольным, если существует разбиение $S^0 \cup S^1 = X$, где S^i устойчиво в H^i, $i = 0, 1$.

В диссертационной работе для задачи поиска всех максимально полных подматриц предложен алгоритм HFINDMCS, который использует в себе новый гиперграфовый подход для генерации полных подматриц. Для задачи поиска всех максимальных индуцированных биклик в графах и гиперграфах предложен алгоритм HFINDMIB, который развивает гиперграфовый подход для генерации индуцированных биклик.

3.3 Алгоритм HFindMCS для решения задачи поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа

Основываясь на определениях теории гиперграфов и $(0,1)$-матриц, опишем алгоритм для нахождения максимально полных подматриц исходной $(0,1)$-матрицы. Для этого понадобится понятие l-уровня $(0,1)$-матрицы.

Определение 3.5. Полным l-уровнем $(0,1)$-матрицы будем называть все ее полные подматрицы с числом строк равным l.

Всякую полную подматрицу инцидентности гиперграфа $H = (X, U)$ с числом строк равным l можно описывать как подгиперграф $H' = (X', U(X'))$, где $X' \subseteq X$ — множество вершин, при этом $|X'| = l$, а $U(X') \subseteq U$ — множество гиперребер. Следовательно, l-уровень в терминах гиперграфов описывается следующей системой множеств $P_l = \{H' = (X', U(X')) : |X'| = l, X' \subseteq X\}$. Также отметим, что гиперграфу H соответствует $(0,1)$-матрица инцидентности I с заданным лексикографическим порядком как для строк, так и для столбцов.

Заметим, что в общем случае всякую $(0,1)$-матрицу можно представить в виде гиперграфа, независимо от ее семантики, и наоборот. Обозначим множество всех максимально полных подматриц как $\mathcal{MCS}$. Идея алгоритма заключается в генерации всех возможных l-уровней $(0,1)$-матрицы и последующем выделении максимально полных подматриц. Иерархия уровней позволяет с легкостью проверять является ли полная подматрица максимальной. Данная идея хорошо реализуется на языке гиперграфов. Схема алгоритма HFINDMCS представлена на рис. 3.3. Описания алгоритма HFINDMCS и его вспомогательной функции $GenerateCombinations(H, l)$ на псевдокоде представлено в алгоритмах 3.1 и 3.2.

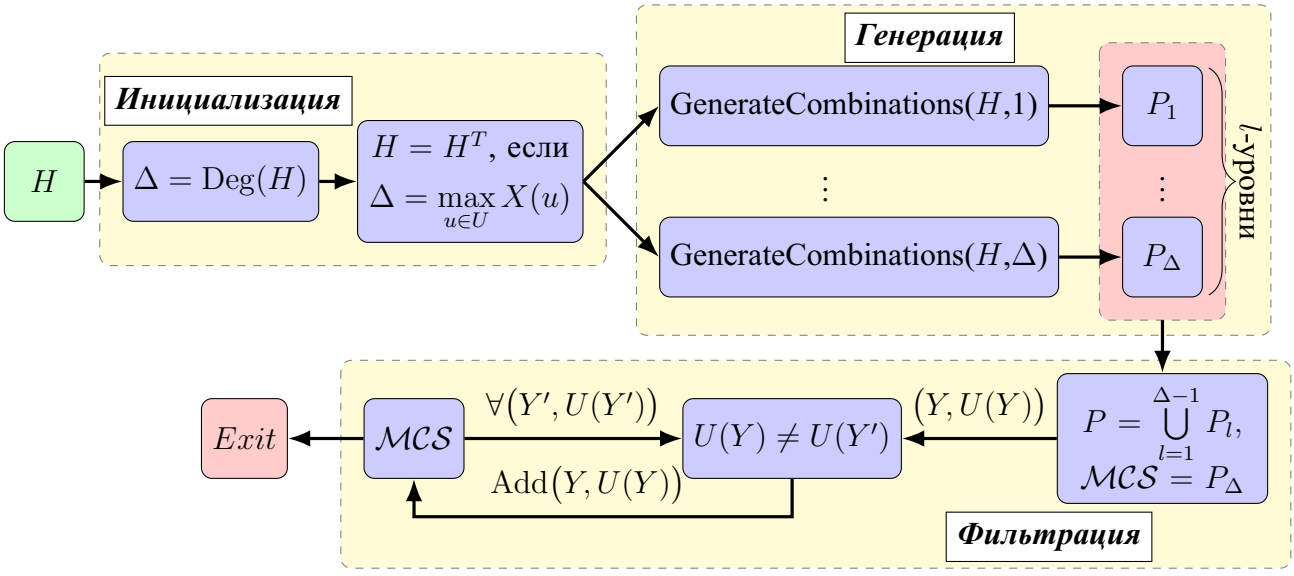


Рисунок 3.3 — Схема алгоритма HFINDMCS

Алгоритм 3.1. Алгоритм HFINDMCS

Вход: гиперграф $H = (X, U)$

Выход: множество $|\mathcal{MCS}|$

Для $l := 1, \ldots, \Delta$

$P_l := GenerateCombinations(H, l)$

Конец цикла

$P := \bigcup\limits_{l=1}^{\Delta} P_l$

$\mathcal{MCS} := \oslash$

Для $\big(Y, U(Y)\big) \in P$

Если $U(Y) \neq U(Y') \colon \forall\big(Y', U(Y')\big) \in \mathcal{MCS}$ **то**

$\mathcal{MCS} := \mathcal{MCS} \cup \big(Y, U(Y)\big)$

Конец условия

Конец цикла

Возвратить $\mathcal{MCS}$

Алгоритм 3.2. Функция $GenerateCombinations(H, l)$

$P_l := \{\oslash\}$

Для $u \in U$

$C_u^l := \big\{X' \colon X' \subseteq X(u), |X'| = l\big\}$

Для $X' \in C_u^l$

$P_l[X'] := P_l[X'] \cup u$

Конец цикла

Конец цикла

Возвратить P_l

Теорема 3.2 (О сложности алгоритма HFINDMCS)**.** *Алгоритм* HFINDMCS *корректно решает задачу нахождения всех максимальных индуцированных биклик гиперграфа* $H = (X, U)$ *со степенью* Δ *за время не превышающее*

$$\mathcal{O}\bigg(\log\big(|MCS|\big) \cdot |U|^2 \cdot \Delta \cdot 2^{\Delta} \cdot \log\big(2^{\Delta} \cdot |U|\big)\bigg).$$

Доказательство. Алгоритм HFINDMCS принимает на вход гиперграф $H = (X, U)$ без кратных гиперребер и вершин. Для простоты изложения будем считать, что максимальная степень гиперребра меньше максимальной степени вершины, т. е. $\max\limits_{u \in U}|X(u)| \leq \max\limits_{x \in X}|U(x)|$. Это условие необходимо для быстродействия алгоритма, в общем случае алгоритм можно применять к любому гиперграфу без кратных гиперребер. Если условие не выполнено, то алгоритм HFINDMCS применяется к двойственному гиперграфу $H^* = (X^*, U^*)$. Далее

будем обозначать максимальную степень гиперребра как $\Delta = \max\limits_{u \in U} |X(u)|$ и называть степенью гиперграфа.

На шагах 1-3 последовательно формируются множества P_l, соответствующие полным l-уровням, где $l = 1, \ldots, \Delta$. Для каждого значения l, соответствующего требуемому уровню $(0,1)$-матрицы, на шаге 2 вызывается функция $GenerateCombinations(H, l)$. Вызываемая функция для каждого гиперребра $u \in U$ выполняет генерацию всех возможных подмножеств вершин $X' \subseteq X(u)$ таких, что $|X'| = l$. Множество всех возможных подмножеств l-уровня обозначим как C_u^l. Каждое сгенерированное множество X' индуцирует исходный гиперграф H в подгиперграф $H' = (X', U')$. Построение множества U' происходит последовательно путем просмотра всех гиперребер $u \in U$. Исходя из этого, множество U' индуцированного подгиперграфа формируется в лексикографическом порядке. Полученный подгиперграф H' однозначно сопоставим с полной подматрицей матрицы I. А множество всех таких подгиперграфов образует полный l-уровень $(0,1)$-матрицы, т. е. множество P_l.

Выполняя данную процедуру для каждого уровня $l = 1, \ldots, \Delta$, получим набор множеств P_l, содержащих подгиперграфы, матрицы инцидентности которых соответствуют всем полным подматрицам матрицы инцидентности I гиперграфа. Очевидно, что полные подматрицы с наибольшим количеством строк равным Δ будут являться максимально полными, поскольку они гарантированно не вложены ни в какие другие полные подматрицы.

При формировании набора множеств P_l процесс генерация сочетаний для всех уровней не превосходит $\tau_1 = \mathcal{O}\left(|U| \cdot 2^{\Delta}\right)$. Действительно, поскольку на каждом уровне необходимо генерировать сочетания для каждого гиперребра и количество уровней $l = 1, \ldots, \Delta$, то

$$|U|C_u^1 + \cdots + |U|C_u^{\Delta} = |U|\left(C_u^1 + \cdots + C_u^{\Delta}\right) = |U|\left(2^{\Delta} - 1\right) = \mathcal{O}\left(|U| \cdot 2^{\Delta}\right).$$

Отметим, что всякое множество $X' \in C_u^l$ является лексикографически упорядоченным и любые X_1', X_2' из C_u^l являются сравнимым. Эти два факта позволяют для хранения и обновления множеств P_l использовать древовидные структуры данных, в частности, красно-черные деревья. Очевидно, что сравнение X_1' и X_2' выполнимо за $\mathcal{O}(\Delta)$, т. к. мощность всякого X' не превосходит Δ.

Также известно [13], что поиск и добавление элементов в красно-черном дереве не превосходит $\tau_2 = \mathcal{O}\left(\Delta \cdot \log\left(|U| \cdot 2^{\Delta}\right)\right)$. Следовательно, выполнение шагов 1-3 алгоритма HFINDMCS требует не более

$$T_1 = \tau_1 \cdot \tau_2 = \mathcal{O}\left(|U| \cdot \Delta \cdot 2^{\Delta} \cdot \log\left(|U| \cdot 2^{\Delta}\right)\right).$$

На шаге 4 алгоритма HFINDMCS полученный на шагах 1-3 набор множеств P_l объединяется в упорядоченное множество P следующим образом: объединение осуществляется в порядке убывания значения уровня l с сохранением у каждого подгипрепрафа лексикографического порядка на множестве вершин и на множестве гиперребер. Следовательно, формирование множества P выполнимо за время равное $T_2 = \mathcal{O}(1)$.

Множество P является допустимым множеством решений задачи поиска всех максимально полных подматриц и соответствующих им гиперграфов. Формирование множества $\mathcal{MCS}$ осуществляется на шагах 5-10, путем просматривания множества P от подгиперграфов с наибольшей мощностью множества вершин до обладающих наименьшей мощностью. Изначально в $\mathcal{MCS}$ добавляются подгиперграфы с числом вершин равным Δ из множества P. Такие подгиперграфы всегда будут соответствовать максимально полным подматрицам. Далее последовательно просматриваются остальные элементы множества P и в множество $\mathcal{MCS}$ добавляются только такие подгиперграфы, множества гиперребер которых ещё не встречались. После этого множество $\mathcal{MCS}$ будет содержать все подгиперграфы, соответствующие максимально полным подматрицам матрицы инцидентности I. Следовательно, алгоритм корректно решает задачу MCSP.

Заметим, что сравнение подгиперграфов по множеству гиперребер требует не более $\mathcal{O}\left(|U|\right)$ времени. Это следует из того, что максимальная мощность множества гиперребер подгиперграфа не превосходит $|U|$. Поскольку по построению все комбинации вершин для подгиперграфов в множестве P различны, то соответствующие им полные подматрицы могут содержаться в других только в случае, когда множество гиперребер совпадает. Это соответствует тому, что полная подматрица содержится в полной подматрице с большим числом строк. Для хранения и обработки множества $\mathcal{MCS}$ в работе так-

же используются красно-черные деревья, что существенно облегчает поиск и добавление новых элементов. Множество P последовательно просматривается, и для каждого элемента выполняется поиск в множестве $\mathcal{MCS}$ за время $T_3 = \mathcal{O}(|U| \cdot \log(|\mathcal{MCS}|))$, аналогичное время требуется для добавления нового элемента. Таким образом, построение множества $\mathcal{MCS}$ требует времени не более, чем

$$T_1 \cdot T_2 \cdot T_3 = \mathcal{O}\left(\log(|MCS|) \cdot |U|^2 \cdot \Delta \cdot 2^{\Delta} \cdot \log\left(2^{\Delta} \cdot |U|\right)\right).$$

□

Искомому множеству $\mathcal{MCS}$ будет соответствовать множество подгипрографов, матрицы инцидентности которых максимально полные. Отметим, что время работы алгоритма HFINDMCS существенно зависит от параметра Δ, который соответствует максимальной степени гиперребра или вершины.

3.4 Алгоритм HFindMIB для решения задачи поиска всех максимальных индуцированных биклик гиперграфа

Пусть задан лексикографический порядок как для множества вершин, так и для для множества гиперребер в гиперграфе H.

В алгоритме рассматривается переход от исходного гиперграфа H, к вершинному графу $L_2(H)$. Матрица смежности $L_2(H)$ представляется в виде гиперграфа $\Phi = (X_\Phi, U_\Phi)$ с квадратной матрицей инцидентности.

Определение 3.6. Подгиперграф Φ' индуцированный множеством вершин $S_0 \subseteq X_\Phi$ и множеством гиперребер $S_1 \subseteq U_\Phi$, удовлетворяющий виду (3.1) и $S_0 \cap S_1 = \oslash$, будем называть бикликой и обозначать (S_0, S_1).

Определение 3.7. Множество всех биклик (S_0, S_1), где $|S_0| = l$, будем называть l-уровнем гиперграфа Φ.

Идея алгоритма заключается в генерации всех возможных биклик гиперграфа для каждого l-уровня и последующем выделении максимальных биклик. Схема алгоритма HFINDMIB представлена на рис. 3.4. Задача MIBGP for Hypergraphs требует найти все максимальные индуцированные биклики

гиперграфа H. Предложенный алгоритм HFINDMIB решает данную задачу в три этапа. На этапе инициализации происходит переход от исходного гиперграфа H к вершинному графу $L_2(H)$. На этапе генерации для матрицы смежности графа $L_2(H)$, предствленной в виде гиперграфа Φ, выполняется генерация l-уровней. На этапе фильтрации происходит очистка всех сгенерированных l-уровней для получения максимальных индуцированных биклик гиперграфа H. Алгоритм HFINDMCS и его вспомогательная функция $GenerateCombinations(H, l)$ представлены на псевдокоде в алгоритмах 3.4 и 3.3. Покажем, что алгоритм находит решение задачи MIBGP for Hypergraphs и оценим трудоемкость алгоритма. Доказательство следующей теоремы носит конструктивный характер, согласно структуре алгоритма.

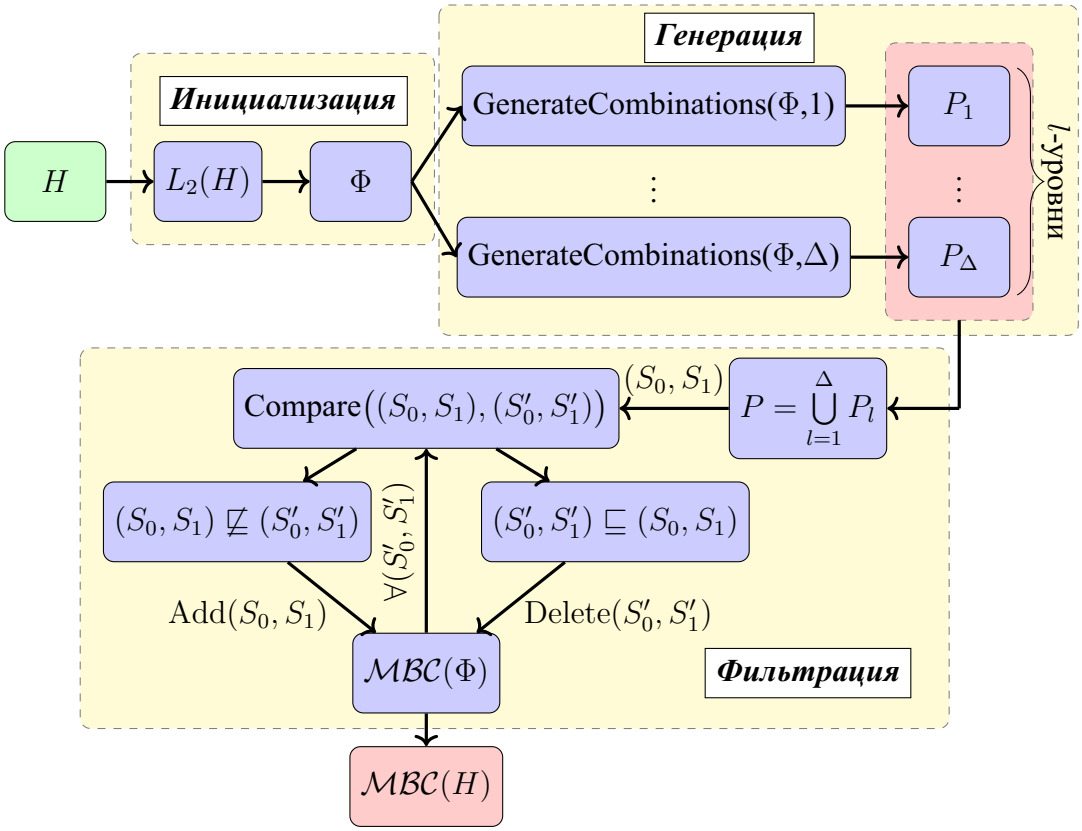


Рисунок 3.4 — Схема алгоритма HFINDMIB

Теорема 3.3 (О сложности алгоритма HFINDMIB)**.** *Алгоритм* HFINDMIB *корректно решает задачу нахождения всех максимальных индуцированных биклик гиперграфа $H = (X, U)$ со степенью Δ за время не превышающее*

$$\mathcal{O}\Big(2^{2\Delta} \cdot \Delta \cdot \big(|\mathcal{MBC}| + \Delta^3 \cdot \log(2^{2\Delta})\big) + |X|^2\Big).$$

Алгоритм 3.3. Функция $GenerateCombinations(\Phi = (X_\Phi, U_\Phi), l)$

$P_l := \oslash$
Для $u \in U_\Phi$
  $C_u^l := \{X' \colon X' \subseteq X(u), |X'| = l\}$
  Для $X' \in C_u^l$
    Если X' является долей **то**
      Для $(S_0, S_1) \in P_l \colon X' = S_0$
        Если $B \cup u$ является долей **то**
          $(S_0, S_1) := (S_0, S_1 \cup u)$
        иначе
          $P_l := (S_0, GetPart(S_1, u))$
        Конец условия
      Конец цикла
    Конец условия
  Конец цикла
Конец цикла
Возвратить P_l

Доказательство. Алгоритм HFINDMIB соостоит из трех основных этапов, последовательно рассмотрим каждый из них.

Этап инициализации. Алгоритм HFINDMIB принимает на вход гиперграф $H = (X, U)$ с максимальной степенью вершины равной Δ. В начале алгоритм строит вершинный граф $L_2(H)$ за время $O\left(|X|^2\right)$ [11]. Для построения систем множеств P_l для заданного гиперграфа H на этапе генерации необходима возможность быстрого обращения к любому подмножеству из l строк матрицы смежности вершинного графа $L_2(H)$. Для организации быстрого доступа предлагается использовать следующий искусственный прием. Для вершинного графа $L_2(H)$ зафиксируем нумерацию множества вершин. Определим новый гиперграф $\Phi = (X_\Phi, U_\Phi)$ следующим образом. Множество вершин X_Φ совпадает с точностью до нумерации с множеством вершин графа $L_2(H)$. Множество гиперребер U_Φ является системой подмножеств из X_Φ и строится в соответствии со столбцами матрицы смежности вершинного графа $L_2(H)$. Заметим, что по построению гиперграф Φ не допускает никакую перенумерацию вершин и гиперребер. Это необходимо для однозначного соответствия между гиперграфом Φ и матрицей смежности вершинного графа $L_2(H)$.

Алгоритм 3.4. Алгоритм нахождения всех максимальных биклик MBC

Вход: гиперграф $H = (X, U)$.

Выход: множество MBC.

Построить граф $L_2(H)$ и представить его в виде гиперграфа Ф
Для $l = 1, \ldots, \Delta$
P_l := GenerateCombinations(Ф,l)
Конец цикла
$P := \bigcup_{l=1}^{\Delta} P_l$
$MBC := \oslash$
Для $(S_0, S_1) \in P$
$Flag$:= true
Для $(S'_0, S'_1) \in MBC$
Проверка вложенности (S_0, S_1) в (S'_0, S'_1)
Если (S'_0, S'_1) вложено в (S_0, S_1) **то**
$MBC := MBC \setminus (S'_0, S'_1)$
Конец условия
Если (S_0, S_1) вложено в (S'_0, S'_1) **то**
$Flag$:= false
Завершить цикл
Конец условия
Конец цикла
Если $Flag$ = true **то**
$MBC := MBC \cup (S_0, S_1)$
Конец условия
Конец цикла
Возвратить MBC

Этап инициализации требует времени не более, чем

$$T_1 = \mathcal{O}\left(|X|^2\right). \tag{3.2}$$

Этап генерации. Результатом данного этапа являются множества всех l-уровней $P_l = \{(S_0, S_1) \colon S_0 \cap S_1 = \oslash\}$, где $l = 1, \ldots, \Delta$. Данные множества содержат все индуцированные биклики исходного гиперграфа H. Генерация осуществляется следующим образом. Формируются множества P_l, соответствующие полным l-уровням, где $l = 1, \ldots, \Delta$. Для каждого значения l выполняется функция $GenerateCombinations(\Phi, l)$. Данная функция для каждого гиперребра $u \in U_\Phi$ выполняет генерацию всех возможных подмножеств

вершин $X' \subseteq X_\Phi(u)$ таких, что $|X'| = l$ и множество X' удовлетворяет виду (3.1), т.е. вершины из X' не смежны друг с другом. Множество всех возможных подмножеств l-уровня гиперребра $u \in U_\Phi$ обозначим как C_u^l. Каждое сгенерированное множество X' рассматривается в качестве доли S_0 биклики. Поскольку гиперграф Φ соответствует матрице смежности графа $L_2(H)$, то всякое $u \in U_\Phi$ можно трактовать как вершину гиперграфа H. Доля S_1 для соответствующих множеств $X' \in C_u^l$ формируется из гиперребер $u \in U_\Phi$, таких что они не противоречат виду матрицы (3.1). Если добавление u к доли S_1 приводит к нарушению вида (3.1), то текущая биклика расщепляется надвое (S_0, S_1) и $(S_0, S_1 \sqcup u)$, где $S_1 \sqcup u$ объединение элементов из S_1 с u таких, что они не являются смежными и отвечают виду (3.1). Таким образом множество P_l будет содержать все возможные индуцированные биклики, у которых $|S_0| = l$.

Оценим сложность этапа генерации. Гиперграф $\Phi = (X_\Phi, U_\Phi)$ соответствует матрице смежности графа $L_2(H)$, поэтому мощность всякого $X_\Phi(u)$, где $u \in U_\Phi$, не превосходит Δ, поскольку максимальная степень вершины исходного гиперграфа H равна Δ. Следовательно, число всевозможных сочетаний подмножеств $X' \in C_u^l$, можно оценить как C_Δ^l. Каждая из долей S_0, S_1 биклик по мощности не превосходит Δ, т. к. максимальная степень вершины равна Δ. Таким образом для любой доли S_0 число возможных долей S_1 можно оценить как 2^Δ. Очевидно, что эта оценка значительно превосходит реальное число возможных индуцированных биклик, в виду условия (3.1). Таким образом, для l-уровня число всех индуцированных биклик не превосходит $C_\Delta^l \cdot 2^\Delta$. Следовательно, число индуцированных биклик среди всех l-уровней можно оценить как

$$2^\Delta \cdot C_\Delta^1 + \cdots + 2^\Delta \cdot C_\Delta^\Delta = 2^\Delta \cdot \left(C_\Delta^1 + \cdots + C_\Delta^\Delta\right) = 2^\Delta \cdot 2^\Delta = \mathcal{O}\left(2^{2\Delta}\right).$$

Отметим, что всякое множество $X' \in C_u^l$ является лексикографически упорядоченным и любые X_1' и X_2' из C_u^l являются сравнимым. Эти два факта позволяют для хранения и обновления множеств P_l использовать древовидные структуры данных, в частности, красно-черные деревья. Очевидно, что сравнение X_1' и X_2' выполнимо за $\mathcal{O}(\Delta)$, т.к. мощность всякого X' не превосходит Δ. А поиск и добавление элементов в красно-черном дереве не превосхо-

дит $\mathcal{O}\Big(\Delta \cdot \log\left(2^{2\Delta}\right)\Big)$ [13]. При этом операция расщепления предполагает проверку вида (3.1), которая займет не более чем $\mathcal{O}\left(\Delta^2\right)$ операций, и нахождение несмежных вершин в доле S_1, что требует не более $\mathcal{O}(\Delta)$. Так, выполнение этапа генерации алгоритма HFINDMIB требует времени не более, чем

$$T_2 = \mathcal{O}\Big(2^{2\Delta} \cdot \Delta^4 \cdot \log\left(2^{2\Delta}\right)\Big). \tag{3.3}$$

Этап фильтрации. Множество всех индуцированных биклик P строится из множеств P_l, которые являются l-уровнями гиперграфа. Как было показано, множества P_l содержат все индуцированные биклики с долями S_0, S_1, причем $|S_0| = l$, а $|S_1| \leq \Delta$. Процесс объединения множеств P_l в множество P осуществляется за $\mathcal{O}(1)$. Этап фильтрации выполняет очистку множества P от избыточных и вложенных индуцированных биклик. Исходя из этапа генерации, в процессе генерируются биклики (S_0, S_1) и (S_0', S_1'), где $S_0 = S_1'$ и $S_1 = S_0'$, т.е. (S_0, S_1) и (S_0', S_1') отражают одну индуцированную биклику. В задаче MIBGP for Hypergraph необходимо найти все максимальные индуцированные биклики, следовательно требуется убрать биклики, которые не являются максимальными. Данный процесс напрямую зависит от мощности выходного множества максимальных индуцированных биклик $\mathcal{MBC}(\Phi)$. Процесс сравнения и определения вложенности происходит в процедуре $Compare\left((S_0, S_1), (S_0', S_1')\right)$. Данная процедура вызывается для каждого элемента из P и сравнивает его с каждым элементом в $\mathcal{MBC}(\Phi)$. Определим операцию $(S_0, S_1) \sqsubseteq (S_0', S_1')$. Если $S_0 \subseteq S_0'$ и $S_1 \subseteq S_1'$, либо $S_1 \subseteq S_0'$ и $S_0 \subseteq S_1'$, то биклика (S_0, S_1) вложена в биклику (S_0', S_1'). Следовательно индуцированная биклика (S_0', S_1') является большей относительно (S_0, S_1). Если индуцированная биклика $(S_0, S_1) \not\sqsubseteq (S_0', S_1')$, где $(S_0', S_1') \in P$, то она является максимальной и добавляется в множество $\mathcal{MBC}(\Phi)$. Такая проверка выполнима за $4 \cdot \Delta$ операций, следовательно требует не более $\mathcal{O}(\Delta)$ времени. На этапе фильтрации требуется отфильтровать $2^{2\Delta}$ элементов множества P и сравнить их не более чем с $|\mathcal{MBC}(\Phi)|$ элементами, что осуществимо за время, не превышающее

$$T_3 = \mathcal{O}\left(2^{2\Delta} \cdot \Delta \cdot |\mathcal{MBC}(\Phi)|\right). \tag{3.4}$$

Согласно теореме 3.1 множество $\mathcal{MBC}(\Phi)$ будет совпадать с множеством максимальных индуцированных биклик исходного гиперграфа H. Процесс фильтрации выполняет сравнение индуцированных биклик друг с другом. После выполнения данного этапа из множества P будут извлечены только максимальные индуцированные биклики, таким образом, алгоритм HFindMIB корректно решает задачу MIBGP for Hypergraphs.

Исходя из сложности каждого этапа (3.2)–(3.4) алгоритм HFINDMIB требует времени не более, чем

$$T_1 + T_2 + T_3 = \mathcal{O}\left(2^{2\Delta} \cdot \Delta \cdot \left(|\mathcal{MBC}| + \Delta^3 \cdot \log\left(2^{2\Delta}\right)\right) + |X|^2\right). \tag{3.5}$$

□

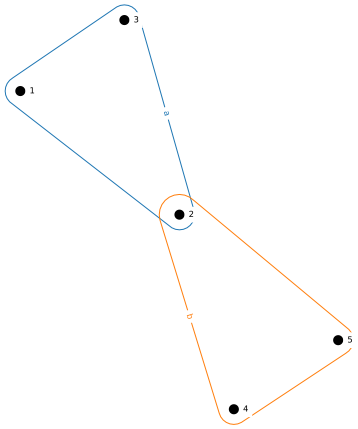
Оценка (3.5), полученная в теореме 3.3, зависит от мощности множества $\mathcal{MBC}$. Это характерная черта для задачи MIBGP for Hypergraphs, которая относится к задачам перечислительного типа.

Проиллюстрируем работу алгоритма на простом гиперграфе. На рис. 3.5 проиллюстрирован этап инициализации алгоритма HFindMIB. Исходный гиперграф H с матрицей инцидентности I представлен на рис. 3.5а).

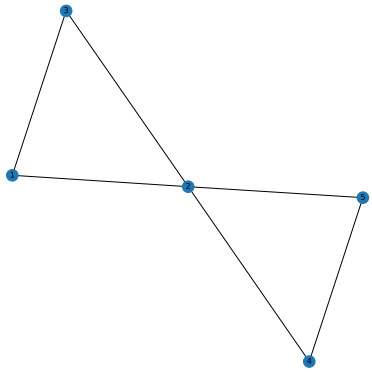
Легко заметить, что максимальными бикликами являются пары $\big(\{1\},\{3\}\big)$, $\big(\{2\},\{1,4\}\big)$, $\big(\{2\},\{1,5\}\big)$, $\big(\{2\},\{3,4\}\big)$, $\big(\{2\},\{3,5\}\big)$, $\big(\{4\},\{5\}\big)$. Для простоты в данном примере будем записывать $\big(\{1,4\},\{2\}\big)$ как $(14,2)$, т. е. $S_0 = \{1,4\}$ и $S_1 = \{2\}$. На рисунке 3.5б) представлен вершинный граф $L_2(H)$ исходного гиперграфа H. В табл. 3.5г) и 3.5д) представлены матрица инцидентности гиперграфа H и матрица смежности вершинного графа $L_2(H)$. Гиперграф Φ, из рис. 3.5в), строится на основе матрицы смежности вершинного графа $L_2(H)$. Для исходного гиперграфа H максимальная степень вершины $\Delta = 4$.

Этапы генерации для каждого Δ представлены табл. 3.2.

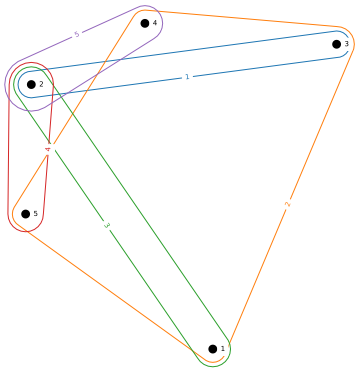
На этапе фильтрации из табл. 3.2, полученной на предыдущем этапе, убираются столбцы 1, 7–9, 11–16, поскольку они являются вложенными в другие индуцированные биклики, либо совпадают с ранее добавленными. Заметим, что биклики $(2,14)$ и $(14,2)$ совпадают. Таким образом, множество $\mathcal{MBC}$ со-



а) гиперграф H



б) вершинный граф $L_2(H)$



в) гиперграф Ф

	a	b
1	1	0
2	1	1
3	1	0
4	0	1
5	0	1

г) матрица инцидентности гиперграфа H

	1	**2**	**3**	**4**	**5**
1	0	1	1	0	0
2	1	0	1	1	1
3	1	1	0	0	0
4	0	1	0	0	1
5	0	1	0	1	0

д) матрица инцидентности гиперграфа Ф

Рисунок 3.5 — Этап инициализации алгоритма HFindMIB

держит максимальные индуцированные биклики $(1, 3)$, $(2, 14)$, $(2, 15)$, $(2, 34)$, $(2, 35)$, $(4, 5)$, соответствующие столбцам 2–6, 10 табл. 3.2.

Таблица 3.2 — Индуцированные билкики для всех уровней, где зеленым цветом выделены максимальные индуцированные билклики для гиперграфа H

Δ	1												2				3	4
№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	—	—
(S_0, S_1)	$(1,2)$	$(1,3)$	$(2,14)$	$(2,15)$	$(2,34)$	$(2,35)$	$(3,1)$	$(3,2)$	$(4,2)$	$(4,5)$	$(5,2)$	$(5,4)$	$(14,2)$	$(15,2)$	$(34,2)$	$(35,2)$		

3.5 Выводы по главе 3

В данной главе предложены алгоритмы HFINDMCS и HFINDMIB для решения задач перечислительного типа. Алгоритм HFINDMCS решает задачу поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа, а алгоритм HFINDMIB решает задачу перечисления всех максимальных индуцированных биклик гиперграфа, соответственно.

Следует отметить следующие особенности разработанных алгоритмов HFINDMCS и HFINDMIB:

1. Для алгоритма HFINDMCS приведена теоретическая сложность в теореме 3.2. Наиболее популярным алгоритмом для решения задачи выделения максимально полных подматриц является алгоритм нахождения всех формальных понятий формального контекста Close-by-One (CbO) [71]. Асимптотическая сложность данного алгоритма оценивается как $\mathcal{O}\left(|\mathcal{MCS}| \cdot |U| \cdot |X|^2\right)$. В общем случае предложенный алгоритм HFINDMCS асимптотически медленнее алгоритма CbO, однако, для наборов данных, организованных определенным образом [19], он будет эффективнее алгоритма CbO. В частности, для квадратных матриц алгоритм HFINDMCS будет асимптотически быстрее алгоритма CbO при выполнении условия $\Big(\Delta \cdot 2^{\Delta} \cdot \log\big(2^{\Delta} \cdot |X|\big)\Big) \leq |X|$. Данное условие может служить рекомендацией по применимости алгоритма HFINDMCS.

2. Алгоритм HFINDMIB основан на доказанной в работе теореме 3.1. Теоретическая сложность алгоритма приведена в теореме 3.3. Полученная в теореме 3.3 оценка, которая зависит от максимальной степени вершины Δ, является завышенной, поскольку часть сочетаний на каждом из генерируемых l-уровней не образуют долю биклики. При этом очевидно, что при достаточно малом значении Δ, алгоритм будет демонстрировать высокую производительность.

Глава 4. Программные средства и результаты их применения

В четвертой главе алгоритмы, предложенные в диссертационной работе, реализованы в виде модулей комплекса программ. Каждый модуль был протестирован на случайных и реальных сетях. Результаты тестирования демонстрируют производительность предложенных алгоритмов. Это решает задачу 4 диссертационного исследования. Основные результаты главы опубликованы в работах [23, 25, 32–34, 82]. В параграфе 4.1 представлено описание комплекса программ, реализующего предложенные алгоритмы. В параграфе 4.2 модифицированный алгоритм ALT сравнивается со своим классическим аналогом и алгоритмом Дейкстры. Сравнение производится на сгененированных графах и реальных сетях из базы данных DIMICS. В параграфе 4.3 алгоритм RevTree приближенного решения задачи о кратчайшем пути в ресурсоограниченной сети сравнивается с программным пакетом ILOG IBM CPLEX. Пакет CPLEX находит точное решение задачи в терминах целочисленного линейного программирования. Приведена соответствующая постановка задачи. Представлены результаты вычислительных экспериментов. В параграфе 4.4 представлены результаты вычислительных экспериментов для алгоритмов HFindMCS и HFindMIB для перечислительных задач на гиперграфах. Алгоритм HFindMCS для нахождения максимально полных $(0,1)$-матриц сравнивается с известным алгоритмом Close-by-One. Для алгоритма HFindMIB проведены эксперименты, демонстрирующие его эффективность на гиперграфах с различными параметрами. В параграфе 4.5 приводится анализ дорожных сетей на основе предложенных алгоритмов. Для модифицированного алгоритма ALT и алгоритма RevTree предложены тепловые карты, которые отражают востребованность тех или иных дуг сети в кратчайших путях. Данные карты могут использоваться для выявления магистральных дорог и узких мест. Для максимальных индуцированных биклик, которые найдены алгоритмом HFindMIB, предложена интерпретация для дорожных сетей. Биклики интерпретируются согласно размеру каждой из долей.

4.1 Комплекс программ, реализующий разработанные алгоритмы

Разработан комплекс программ, реализующий предложенные алгоритмы, для проверки их результативности на случайных графах и гиперграфах и на реальных данных применительно к дорожным сетям. Комплекс состоит из четырех модулей, представленных в табл. 4.1.

Таблица 4.1 — Комплекс программ, реализующий предложенные алгоритмы

Название модуля	Назначение модуля
ModALT	Реализует модифицированный алгоритм ALT. Находит решение задачи о кратчайшем пути в нестационарной сети.
RevTree	Реализует алгоритм RevTree. Находит решение задачи о кратчайшем пути в ресурсоограниченной сети.
HFindMCS	Реализует алгоритм HFindMIB. Находит все максимально полные подматрицы $(0,1)$-матрицы инцидентности сети.
HFindMIB	Реализует алгоритм HFindMCS. Находит все максимальные индуцированные биклики сети.

Каждый модуль протестирован на сгенерированных и реальных данных. Модули ModALT, RevTree и HFindMIB использовались для анализа реальных дорожных сетей городов.

4.2 Анализ результативности модифицированного алгоритма ALT

Для оценки результативности модифицированного алгоритма ALT проведены вычислительные эксперименты на случайно сгенерированных графах и дорожных сетях, представленных в базе данных DIMACS [48]. Последовательности запросов σ выбирались случайным образом, при этом $|\sigma| = 500$. Вычислительные эксперименты выполнялись на компьютере с процессором Intel® Core ™i7-720QM Processor (6M Cache, 1.60 GHz) и ОЗУ размером 4 ГБ.

Модифицированный алгоритм ALT сравнивался с алгоритмом Дейкстры и классическим алгоритмом ALT. Напомним, что классический алгоритм ALT предусматривает лишь однократную расстановку ориентиров случайным об-

разом. При сравнении использовался классический алгоритм ALT с эвристикой H_1, описанной в разделе 2.2. Алгоритмы сравнивались по следующим показателям: C_1 — коэффициент сокращения пространства поиска, равный отношению величин $|V_{avg}|$ для сравниваемых алгоритмов; C_2 — коэффициент ускорения, определяемый как отношение времени работы сравниваемых между собой алгоритмов. Заметим, что соответствующие характеристики ($|V_{avg}|$ и время работы) модифицированного алгоритма ALT определяют знаменатель этих отношений. Опираясь на выводы по главе 1 диссертационной работы, выбраны следующие значения параметров эвристики AdaHeuris: число ориентиров $K = 12$, период обновления ориентиров $\Delta = 30$.

Результаты сравнения модифицированного алгоритма ALT с алгоритмом Дейкстры и классическим алгоритмом ALT приведены в таблицах 4.2 и 4.3 соответственно.

Таблица 4.2 — Результаты сравнения модифицированного алгоритма ALT и алгоритма Дейкстры

$G = (V, E)$	$\|V\|$	$\|E\|$	C_1	C_2
Rome99	3353	8870	1,12	1,07
LineGraph	10001	20000	1,03	1,20
GridGraph	62500	498000	1,20	2,19
AK	69082	157662	1,05	1,12
VT	97975	216628	1,08	1,14
CT	153011	375310	1,14	1,30

Таблица 4.3 — Результаты сравнения модифицированного алгоритма ALT и классического алгоритма ALT

$G = (V, E)$	$\|V\|$	$\|E\|$	C_1	C_2
Rome99	3353	8870	1,01	0,97
LineGraph	10001	20000	1,01	1,26
GridGraph	62500	498000	1,05	1,17
AK	69082	157662	1,01	1,05
VT	97975	216628	1,00	1,12
CT	153011	375310	1,05	1,42

Из 6 графов, представленных в таблицах 4.2 и 4.3, графы LineGraph и

GridGraph сгенерированы случайным образом, а все остальные взяты из базы данных DIMACS. Для каждого графа $G = (V, E)$ указаны число вершин $|V|$ и число дуг $|E|$. Графы расположены в порядке возрастания $|V|$.

Из таблиц 4.2 и 4.3 видно, что модифицированный алгоритм ALT дает незначительное сокращение пространства поиска. Однако при выбранных параметрах K и Δ наблюдается уменьшение времени работы данного алгоритма по сравнению с алгоритмом Дейкстры и классическим алгоритмом ALT. Причем результативность по времени тем выше, чем больше $|V|$.

По результатам экспериментов можно сделать вывод, что предложенная модификация алгоритма ALT по быстродействию не уступает алгоритму Дейкстры и своему классическому аналогу, а при соответствующей настройке параметров K и Δ эвристики ADAHEURIS, превосходит их. Такая настройка всегда может быть выполнена с учетом особенностей рассматриваемого графа и возможных последовательностей σ.

4.3 Анализ результативности алгоритма RevTree

Для оценки результативности алгоритма RevTree проведены вычислительные эксперименты на компьютере с процессором Intel Core i7-7700K Processor (8 MB Cache, 3,60 ГГц) и ОЗУ объёмом 16 Гбайт. Осуществлялось сравнение программной реализации алгоритма RevTree и пакета IBM ILOG CPLEX [64] по времени работы, числу выполненных запросов, точности найденного решения. Заметим, что обе программы находят решение задачи RCSP, если множество допустимых решений не пусто, при этом CPLEX находит оптимальное (точное) решение. Отметим, что пакет CPLEX решает задачу RCSP в терминах целочисленного линейного программирования.

Приведем постановку задачи о ресурсоограниченном кратчайшем пути в терминах целочисленного линейного программирования [63]. Пусть булева переменная x_e соответствует каждой дуге $e \in E$ графа G и показывает, проходит ли искомый путь через эту дугу или нет. Задача RCSP в терминах целочисленного линейного программирования формулируется следующим образом

$$\sum_{e \in E} w(e) x_e \to \min; \tag{4.1}$$

$$\begin{cases} \sum\limits_{e\in\Gamma(i)} x_e - \sum\limits_{e\in\Gamma^{-1}(i)} x_e = \begin{cases} 1, & i = s; \\ 0, & i \in V \setminus \{s, d\}; \\ -1, & i = d; \end{cases} & (4.2) \\ \sum\limits_{e\in E} r_i(e) x_e \le R_i, \quad i = 1, \ldots, k; & (4.3) \\ x_e \in \{0, 1\}, \quad e \in E. & (4.4) \end{cases}$$

Множества $\Gamma(i)$ и $\Gamma^{-1}(i)$ содержат в себе все исходящие из вершины i и входящие в это вершину дуги, соответственно.

Эксперименты проводились на случайно сгенерированных графах G_1–G_3 для последовательности из 1000 случайно сгенерированных (s, d)-запросов (табл. 4.4). Для случайной генерации графов применялся метод Ваксмена с параметрами $\alpha = 0{,}15$, $\beta = 0{,}25$, который традиционно используется для генерации графов топологически схожих с реальными компьютерными сетями [78]. Запрос считался выполненным, если для него было найдено допустимое решение задачи RCSP. Результаты экспериментов представлены в табл. 4.5.

Таблица 4.4 — Размерность и параметры задачи RCSP

$G = (V, E)$	$\lvert V\rvert$	$\lvert E\rvert$	$\lambda_{\max}$	$\lambda_{\min}$	ε
G_1	500	3923	0,75	0,6	0,25
G_2	1000	16345	0,75	0,6	0,25
G_3	1500	36655	0,75	0,6	0,25

Таблица 4.5 — Результаты сравнения алгоритма RevTree и пакета CPLEX

Название графа	CPLEX		RevTree		
	Время обработки серии запросов, с	Число выполненных запросов	Время обработки серии запросов, с	Число выполненных запросов, из них для q найдено приближенное решение	
				всего	q
G_1	428,265	910	2,69	910	0
G_2	1607,58	967	47,524	967	0
G_3	7571,22	676	94,217	676	0

Согласно этим результатам, алгоритм RevTree для рассматриваемых гра-

фов находит столько же допустимых решений, что и CPLEX, при этом они совпадают и являются оптимальными. Во всех случаях алгоритм RevTree работает значительно быстрее пакета программ CPLEX.

4.4 Анализ результативности алгоритмов HFindMCS и HFindMIB

Вычислительные эксперименты для алгоритмов HFindMCS и HFindMIB выполнялись на компьютере с процессором AMD Ryzen 5 3600 6-Core Processor 3,60 ГГц и ОЗУ объемом 16 Гбайт.

Для оценки результативности решения задачи MCSP for Hypergraphs алгоритмом HFindMCS проведены вычислительные эксперименты. Алгоритм HFindMCS сравнивался с алгоритмом Close-by-One на сгенерированных гиперграфах $H = (X, U)$ с различным числом вершин $|X|$ и гиперребер $|U|$, а также различной максимальной степенью Δ. Заметим, что исходя из результатов, полученных в главе 3, для квадратных матриц алгоритм HFindMCS будет асимптотически быстрее алгоритма Close-by-One при выполнении условия $\Big(\Delta \cdot 2^{\Delta} \cdot \log\big(2^{\Delta} \cdot |X|\big)\Big) \leq |X|$, поэтому вычислительные эксперименты проводились именно для подобных матриц. Исследовалась зависимость времени работы алгоритма HFindMCS от мощности множества $|\mathcal{MCS}|$, а также время работы алгоритмов HFindMCS и Close-by-One. Для оценки результативности решения задачи MIBGP for Hypergraphs алгоритмом HFindMIB, эксперименты проводились на гиперграфах $H = (X, U)$ с различным числом вершин $|X|$ и гиперребер $|U|$, а также различной максимальной степенью вершины Δ. Результаты вычислительных экспериментов по алгоритмам HFindMCS и HFindMIB представлены в табл. 4.6 и 4.7.

Как видно из табл. 4.6, алгоритм HFindMCS находит решение задачи MCSP for Hypergraphs быстрее алгоритма Close-by-One для гиперграфов с матрицей инцидентности близкой к квадратной и малой степенью. По результатам из табл. 4.7 можно сделать вывод, что время выполнения алгоритма HFindMIB существенно зависит от мощности множества $\mathcal{MBC}$. Это свойственно для задачи поиска всех максимальных индуцированных биклик, поскольку их число может экспоненциально зависеть от размеров гиперграфа.

Таблица 4.6 — Результаты алгоритма HFINDMCS

Δ	$\lvert X\rvert$	$\lvert U\rvert$	$\lvert\mathcal{MBC}\rvert$	t, сек	t_{CbO}, сек
5	100	100	276	0,002	0,005
	1000	1000	2055	0,012	0,301
	10000	9904	19609	0,145	26,693
	25000	24540	48220	0,421	177,749
10	100	100	1199	0,041	0,022
	1000	1000	3786	0,422	0,500
	10000	9999	22037	4,938	36,292
	25000	24989	52501	13,571	224,405
13	100	100	2570	0,339	0,046
	1000	1000	7056	3,521	0,871
	10000	10000	26102	38,091	40,194
	25000	24999	57420	497,621	—

Таблица 4.7 — Результаты алгоритма HFINDMIB

Δ	$\lvert X\rvert$	$\lvert U\rvert$	$\lvert\mathcal{MBC}\rvert$	t, сек
3	100	88,5	192	0,027
	500	438,8	965,1	0,446
	1000	870,1	1943,3	1,803
	2500	2186,6	4845,4	12,726
5	100	78,9	443,8	0,101
	500	371,5	2304,3	0,260
	1000	732,4	4624,1	11,321
	2500	1830	11615,9	74,750
7	100	73,5	818,2	0,367
	500	330	4285,9	9,904
	1000	648,3	8633,6	55,452
	2500	1586,5	21813,7	239,831

4.5 Анализ дорожных сетей

Предлагаемые в работе алгоримы REVTREE, HFINDMIB и модифицированный алгоритм ALT применялись для анализа дорожных сетей. Дорожные сети были взяты из веб-картографического проекта OpenStreetMap с помощью пакета OSMnx для Python [41]. В качестве рассматриваемых сетей были взяты дорожные карты следующих городов: Красноярск (KJA), Томск (TOF),

Новосибирск (NSK), Баку (GYD). Для модифицированного алгоритма ALT и алгоритма RevTree были сгенерированы 1000 запросов для каждой сети. Алгоритм HFindMIB применялся к неориентированным версиям представленных сетей. Результаты работы каждого алгоритма представлены в табл. 4.8. В табл. 4.9 представлено число максимальных индуцированных биклик разного размера для каждой сети, найденные алгоритмом HFindMIB.

Таблица 4.8 — Результаты работы алгоритмов ALT, RevTree, HFindMIB

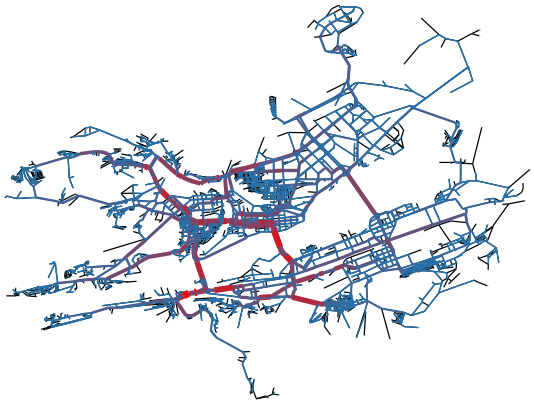
Дорожная сеть	n	m	ALT	RevTree		HFindMIB		
			Время, с.	Доля выполнынных запросов	Время, с.	Δ	$\mathcal{MBC}$	Время, с.
TOF	3840	9639	8,151	0,483	1,708	5	4084	26,527
KJA	4362	6005	9,030	0,849	7,286	5	4823	36,898
NSK	7357	19106	27,838	0,680	12,212	6	7927	103,115
GYD	12458	28936	69,052	0,539	15,998	6	11844	252,244

Таблица 4.9 — Число максимальных индуцированных биклик разного размера

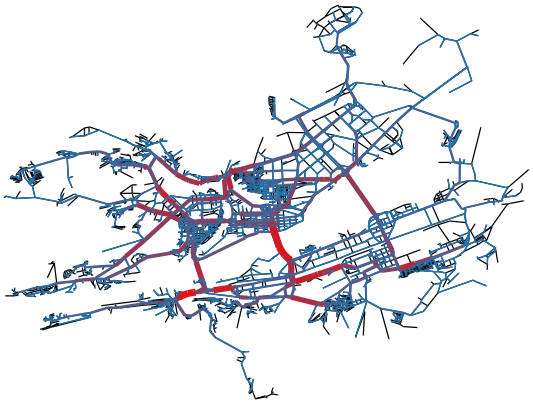
$(\lvert S_0\rvert, \lvert S_1\rvert)$	TOF	KJA	NSK	GYD
$(1, 1)$	1	3	5	2
$(2, 1)$	955	1454	1612	2247
$(2, 2)$	589	535	1204	1154
$(3, 1)$	1981	2315	3886	7217
$(3, 2)$	2	—	1	—
$(4, 1)$	533	514	1215	1215
$(5, 1)$	3	2	4	9

На рис. 4.1а) представлена тепловая карта сети KJA для решений задачи TDSP, где дорога обозначена красным цветом в случае, если она была более востребована в процессе маршрутизации и синим цветом в обратном случае, черным цветом обозначены ребра, которые не встретились ни в одном кратчайшем пути. Аналогичная тепловая карта представлена по результатам решения задачи RCSP для сети KJA на рис. 4.1б). Видно, что тепловые карты на рис. 4.1а) и 4.1б) имеют общие участки дорог. Это говорит о том, что данные дороги одинаково востребованы как в сетях, зависящих от времени, так

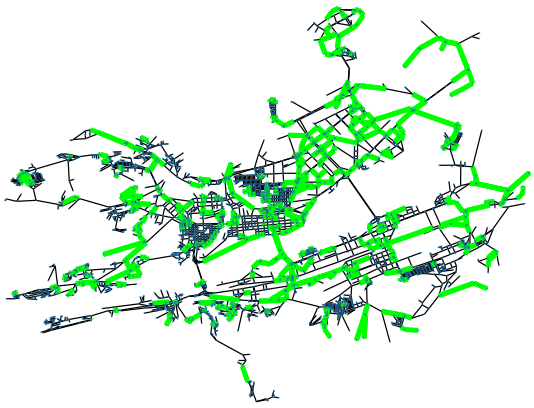
и в ресурсоограниченных. Таким образом, можно сделать вывод, что данные дороги носят магистральный характер.



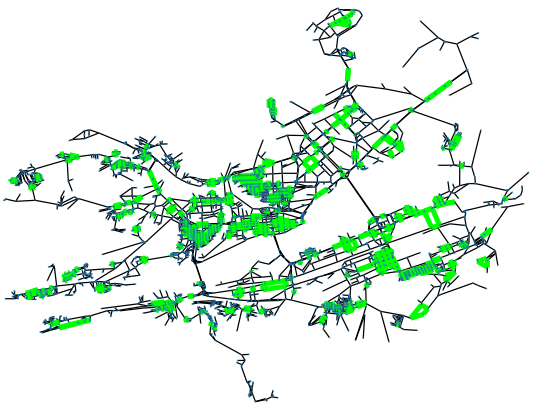
а) тепловая карта сети KJA по результатам решения задачи TDSP



б) тепловая карта сети KJA по результатам решения задачи RCSP



в) карта сети KJA с бикликами размера $(2, 1)$



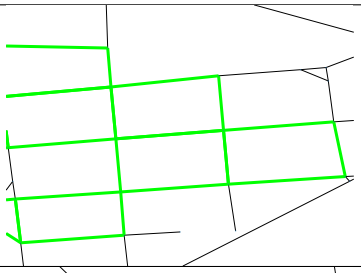
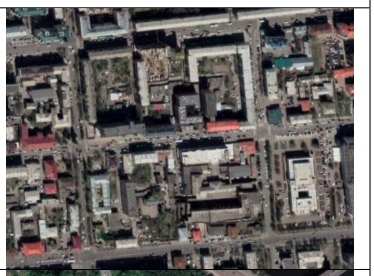
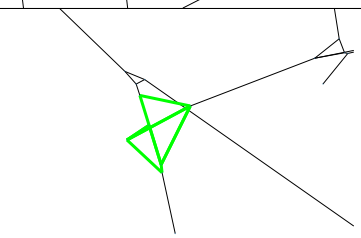
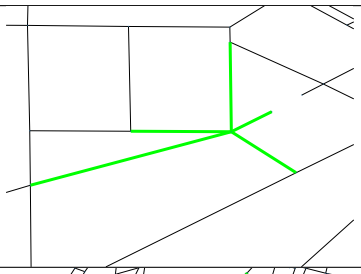
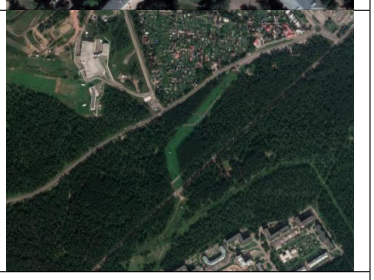
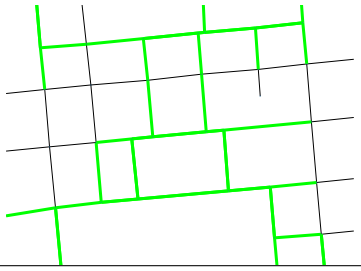
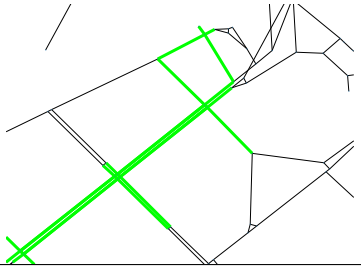
г) карта сети KJA с бикликами размера $(2, 2)$

Рисунок 4.1 — Анализ дорожной сети KJA на основе найденных решений

Сопоставим некоторым бикликам реальные фрагменты карты. Фрагменты карт взяты с сервиса «Google Maps» [56]. В табл. 4.10 представлены биклики и фрагменты снимков со спутника.

На основе полученных алгоритмом HFINDMIB результатов, сделаны выводы о возможной интерпретации элементов дорожной сети согласно размеру биклики. Пусть (S_0, S_1) выделенная максимальная индуцированная биклика с долями S_0 и S_1. В случае, когда $|S_0| = |S_1| = 2$, то такая биклика может быть интерпретирована как городской квартал. Биклики с $|S_0| = 1$ и $|S_1| \geq 3$ интерпретируются как перекрестки или дорожные кольца. Поскольку все найденные биклики являются максимальными, то при $|S_0| = 1$ и $|S_1| = 2$ биклика

Таблица 4.10 — Биклики, найденные в сетях, и соответствующие им места снятые со спутника

Биклика	Координаты	Биклика в сети	Снимок со спутника
$(2, 2)$, городской квартал	N56.012757°, E92.883504°		
$(3, 2)$, дорожная развязка	N56.449532°, E84.924983°		
$(5, 1)$, перекресток из пяти дорог	N56.460555°, E84.981494°		
$(2, 1)$, протяженные дороги	N56.001779°, E92.762203°		
$(3, 1)$, перекресток с тремя дорогами	N56.011022°, E92.852440°		
$(4, 1)$, перекресток с четырьмя дорогами	N56.068275°, E92.932698°		

будет отображать протяженную или объездную дорогу. Большой интерес для предобработки представляют биклики имеющие «гамачные» структуры, т. е.

когда $|S_0| = 2$ и $|S_1| > 2$, поскольку такие структуры могут быть использованы для уменьшения размеров исходной сети, путем стягивания одной из долей биклики в одну вершину. Биклики с долями $|S_0| = 1$ и $|S_1| = 1$ могут означать отклонения в дорожной сети, либо часть тупиковой закольцованной дороги. Покажем это на примере сети KJA. Максимальные индуцированные биклики размера $(2, 1)$ могут отображать протяженные или объездные дороги. Такие биклики изображены на рис. 4.1в). Максимальные индуцированные биклики размера $(2, 2)$ чаще всего соответствуют городским кварталам, что легко видеть на рис. 4.1г).

Таким образом, на основе решений, найденных алгоритмами RevTree, HFindMIB и модифицированным алгоритмом ALT, выполнен анализ дорожной сети KJA, что отражено на рис. 4.1 и табл. 4.10. Основываясь на информации о наиболее востребованных ребрах в сети можно выполнять реорганизацию дорожной сети и моделировать новые объездные дороги.

4.6 Выводы по главе 4

В данной главе представлены результаты вычислительных экспериментов для предложенных алгоритмов, которые демонстрируют их эффективность.

Вычислительные эксперименты выявили следующие особенности предложенных алгоритмов:

1. Модифицированный алгоритм ALT продемонстрировал высокую эффективность относительно алгоритма Дейкстры и своего классического аналога на реальных сетях. Однако следует отметить, что настройка параметров K и Δ эвристики AdaHeuris для каждой конкретной сети может существенно увеличить эффективность модифицированного алгоритма ALT.

2. Приближенный алгоритм RevTree показал высокую эффективность по времени в сравнении с пакетом ILOG IBM CPLEX. При этом стоит отметить, что количество точно найденных решений совпадает для этих алгоритмов. Достоинство алгоритма RevTree в его полиномиальной сложности и возможности оценить точность находимого решения для сети до непосредственного применения алгоритма.

3. Алгоритм HFINDMCS относительно известного алгоритма Close-by-One продемонстрировал хорошую эффективность на гиперграфах с ограниченной степенью вершины Δ. Как следует из рекомендации по использованию алгоритма HFINDMCS, он превосходит алгоритм Close-by-One на квадратных матрицах начиная с некоторого значения $|X|$ для фиксированного значения Δ.

4. Алгоритм HFINDMIB применялся к сгенерированным гиперграфам с ограниченной степенью вершины Δ. Учитывая сложность задачи поиска всех максимальных индуцированных биклик гиперграфа, алгоритм HFINDMIB способен решать ее за реальное время. Это демонстрируют представленные результаты вычислительных экспериментов.

5. Алгоритм RevTree и модифицированный алгоритм ALT используют накопленные данные о загруженности дорожной сети в разное время суток, чтобы строить оптимальный маршрут с учетом этих данных. Получение кратчайших путей для серии запросов позволяет находить наиболее востребованные дороги в дорожной сети. При этом можно использовать модельную дорожную сеть с добавлением новых дорог или перекрытием существующих, чтобы анализировать последствия таких решений. Алгоритм HFindMIB для нахождения всех максимальных индуцированных биклик позволяет выделять в дорожной сети биклики, которые можно интерпретировать как отдельные дорожные структуры (перекрестки, протяженные дороги, городские кварталы и пр.), тем самым позволяет оценивать скопление тех или иных структур в частях сети. Данная информация, вместе с результатами предыдущих алгоритмов, может использоваться для выделения наиболее загруженные участков сети, согласно исследуемой структуре.

ЗАКЛЮЧЕНИЕ

1. Разработан и теоретически обоснован модифицированный алгоритм ALT для решения задачи поиска кратчайшего пути в нестационарной метрической сети, удовлетворяющей условию FIFO (теорема 1.1). В алгоритме используется новая графовая модель с двумя весами. С использованием этой модели вычисляются потенциальные функции для алгоритма ALT.
2. Разработан и теоретически обоснован приближенный алгоритм RevTree для решения задачи поиска кратчайшего пути в ресурсоограниченной сети с одним ресурсом (теоремы 2.1, 2.2). Алгоритм позволяет определить точность найденного решения исходя из параметров сети.
3. Сформулирована и доказана теорема об эквивалентности индуцированных двудольных подгиперграфов гиперграфа и двудольных подграфах вершинного графа гиперграфа (теорема 3.1).
4. Разработан и теоретически обоснован алгоритм HFindMCS для поиска всех максимально полных подматриц $(0,1)$-матрицы инцидентности гиперграфа (теорема 3.2). Алгоритм HFindMCS позволяет эффективно находить решение для гиперграфа с малой степенью и квадратной разреженной $(0,1)$-матрицей инцидентности.
5. Разработан и теоретически обоснован алгоритм HFindMIB для поиска всех максимальных индуцированных биклик гиперграфа (теорема 3.3). Алгоритм основан на теореме 3.1 и использует метод генерации максимальных биклик для каждого l-уровня гиперграфа.
6. Создан комплекс программ, реализующий разработанные алгоритмы, для проверки их результативности на случайных графах и гиперграфах и на реальных данных применительно к дорожным сетям.

Список литературы

1. Блюмин С. Л. Полные гиперграфы. Спектры лапласианов. Мультиагентные системы / С. Л. Блюмин // Управление большими системами. — 2010. — №30. — С. 5–23.
2. Быкова В. В. Адаптивное размещение ориентиров при маршрутизации в нестационарных сетях / В. В. Быкова, А. А. Солдатенко // Supplementary Proceedings of the 9th International Conference on Discrete Optimization and Operations Research and Scientific School (DOOR 2016) — Vladivostok, Russia, September 19-23, 2016. — Published online on the CEUR-Workshop web site, V. 1623. — P. 367–372.
3. Быкова В. В. Декомпозиционный подход к исследованию формальных контекстов / В. В. Быкова, Ч. М. Монгуш // Прикладная дискретная математика. — 2019. — № 44. — С. 113–126.
4. Быкова В. В. Маршрутизация по ориентирам в нестационарных сетях / В. В. Быкова, А. А. Солдатенко // Доклады Седьмой Международной научной конференции «Танаевские чтения», Минск, Беларусь. — 2016. — С. 40–44.
5. Быкова В. В. Об оптимальной маршрутизации в мультисервисных телекоммуникационных сетях / В. В. Быкова, А. А. Солдатенко // Optimization Problems and Their Applications (OPTA-2018): тезисы докладов VII Международной конференции: памяти проф. А. А. Колоколова — Омск: Изд-во Ом. гос. ун-та, 2018. — С. 26.
6. Быкова В. В. Об оценке ресурсных возможностей мультисервисных сетей / В. В. Быкова, А. А. Солдатенко // Информационные технологии и математическое моделирование (ИТММ-2018): Материалы XVII Международной конференции имени А.Ф. Терпугова. — Томск: Изд-во НТЛ, 2018. — С. 230–235.
7. Быкова В. В. Оптимальная маршрутизация по ориентирам в нестационарных сетях / В. В. Быкова, А. А. Солдатенко // Прикладная дискретная математика. — 2017. — № 37. — С. 114–123.

8. Гимади Э. Х. Математические модели и методы принятия решений / Э. Х. Гимади, Н. И. Глебов. — Новосибирск: Издательство НГУ, 2008. — 162 с.
9. Гэри М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон // М.: Мир, 1982. — 416 с.
10. Емеличев В. А. Лекции по теории графов / В. А. Емеличев, О. И. Мельников, В. И. Сарванов, Р. И. Тышкевич. — М.: Книжный дом «Либроком», 2012. — 390 с.
11. Зыков А. А. Гиперграфы / А. А. Зыков // УМН. — 1974. — Т. 29. — №6(180). — С. 89–154.
12. Карим П. Х. Численные исследования пропускной способности транспортного протокола с механизмом прямой коррекции ошибок в межсегментном пространстве / П. Х. Карим, П. А. Михеев, В. В. Поддубный, С. П Сущенко // Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика. — 2020. — № 50. — С. 89–96.
13. Кормен Т. Х. Алгоритмы. Построение и анализ / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн. — М.: Вильямс, 2016. — 1328 с.
14. Корте Ф. Комбинаторная оптимизация. Теория и алгоритмы / Ф. Корте, Й. Фиген // М.:МЦНМО. — 720 с.
15. Кофман А. Введение в прикладную комбинаторику / А. Кофман // М.: Наука, 1975. — 480 с.
16. Кузнецов С. О. Интерпретация на графах и сложностные характеристики задач поиска закономерностей определенного вида / С. О. Кузнецов // Научно-техническая информация (НТИ). — 1989. — Сер. 2. — № 1. — С. 23–28.
17. Маркус М. Обзор по теории матриц и матричных неравенств / М. Маркус, Х. Минк // М.: Наука, 1972. — 232 с.
18. Мейер Д. Теория реляционных баз данных / Д. Мейер // М.: Мир, 1987. — 608 с.
19. Миллер Р. Последовательные и параллельные алгоритмы: Общий подход / Р. Миллер, Л. Боксер // М.: БИНОМ. Лаборатория знаний, 2009. — 406 с.

20. Попов В. Б. Экстремальная нумерация вершин гиперграфа и задача объектно-признаковой кластеризации / В. Б. Попов // Динамические системы. — 2010. — №28. — С. 99–112.
21. Рассел С. Искусственный интеллект. Современный подход / С. Рассел, П. Норвиг. — М.: Вильямс, 2007. — 1410 с.
22. Семенова Д. В. Анализ формальных понятий на языке гиперграфов / Д. В. Семенова, А. А. Солдатенко // Математика, ее приложения и математическое образование МПМО'20 Материалы VII Международной конференции. — Улан-Удэ: Изд-во ВСГУТУ. — 2020. — С. 186–188.
23. Солдатенко А. А. Адаптивный алгоритм поиска оптимального маршрута в нестационарной сети / А. А. Солдатенко // Программные продукты и системы. — 2018. — № 2. — С. 321–329.
24. Солдатенко А. А. Алгоритм оптимальной маршрутизации в мультисервисных телекоммуникационных сетях / А. А. Солдатенко // Прикладная дискретная математика. Приложение. — 2018. — № 11. — С. 122–127.
25. Солдатенко А. А. Алгоритм HGFC нахождения формальных понятий / А. А. Солдатенко, Д. В. Семенова // Информационные технологии и математическое моделирование (ИТММ-2020) материалы XIX Международной конференции имени А. Ф. Терпугова. — Томск: Изд-во НТЛ. — 2020. — С. 478–482.
26. Солдатенко А. А. Верхнее оценивание стоимости оптимального маршрута в сети с ограничением / А. А. Солдатенко // Информационные технологии и математическое моделирование (ИТММ-2019) материалы XVIII Международной конференции имени А. Ф. Терпугова. — Томск: Изд-во НТЛ. — 2019. — Часть 2. — С. 47–52.
27. Солдатенко А. А. Двухфазный алгоритм маршрутизации в нестационарных сетях / А. А. Солдатенко // Прикладная дискретная математика. Приложение. — 2017. — № 10. — С. 168—171.
28. Солдатенко А. А. О нахождении максимально полных подматриц и их связи с бикликами в гиперграфе / А. А. Солдатенко, Д. В. Семенова // Вест-

ник Томского государственного университета. Управление, вычислительная техника и информатика. — 2021. — № 56. — С. 90—99.

29. Солдатенко А. А. О решении задачи TDSP двухфазным алгоритмом на больших сетях / А. А. Солдатенко // Материалы республиканской научно-практической конференции «СТАТИСТИКА и ее применения - 2017», Ташкент. — Ташкент, НУУз, 2017. — С. 84–91.
30. Солдатенко А. А. Полиномиальный алгоритм поиска приближенного решения в графе с ограничением / А. А. Солдатенко // Труды республиканской научно-практической конференции СТАТИСТИКА и ее применения-2019. — Ташкент, Филиал МГУ. — 2019. — С. 178–183.
31. Солдатенко А. А. Приближенный алгоритм поиска оптимального маршрута в сети с ограничением / А. А. Солдатенко // Прикладная дискретная математика. Приложение. — 2019. — № 12. — С. 186–191.
32. Солдатенко А. А. Программа HFindMCS выделения максимально полных подматриц $(0,1)$-матрицы. Свидетельство о государственной регистрации программы для ЭВМ № 2021662444. Зарегистрировано в Реестре программ для ЭВМ от 11 августа 2021 г.
33. Солдатенко А. А. Программа HFindMIB выделения максимальных индуцированных биклик гиперграфа. Свидетельство о государственной регистрации программы для ЭВМ № 2021662445. Зарегистрировано в Реестре программ для ЭВМ от 11 августа 2021 г.
34. Солдатенко А. А. Программа RevTree для поиска ресурсоограниченного пути в сети. Свидетельство о государственной регистрации программы для ЭВМ № 2019616547. / А. А. Солдатенко. Зарегистрировано в Реестре программ для ЭВМ от 24 мая 2019 г.
35. Тараканов В. Е. Комбинаторные задачи и (0, 1)-матрицы / В. Е. Тараканов // М.: Наука, 1985. — 195 с.
36. Харари Ф. Теория графов / Ф. Харари // URSS, 2018. — 304 с.
37. Acuna V. Solving the maximum edge biclique packing problem on unbalanced bipartite graphs / V. Acuna, C. E. Ferreira, A. S. Freire, E. Moreno // Discrete Applied Mathematics. — 2014. — V. 164, Part 1. — P. 2–12.

38. Alexe G. Consensus algorithms for the generation of all maximal bicliques / G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer, B. Simeone // Discrete Applied Mathematics. — 2004. — V. 145. — P. 11–21.
39. Beineke L. Topics in Algebraic Graph Theory / L. Beineke, R. J. Wilson // Mathematical Sciences Faculty Publications, 2008.
40. Bejerano Y. Algorithms for computing QoS paths with restoration / Y. Bejerano, Y. Breitbart, A. Orda, R. Rastogi, A. Sprintson // IEEE/ACM Transactions on Networking. — 2005. — V. 13. — №3. — P. 648–661.
41. Boeing G. OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. / G. Boeing // Computers, Environment and Urban Systems. — 2017. — V. 65. — P. 126–139.
42. Bretto A. Hypergraph theory: an introduction / A. Bretto // Springer, 2013.
43. Cooke K. L. The shortest route through a network with time-dependent internodal transit times / K. L. Cooke, E. Halsey // Journal of Mathematical Analysis and Applications. — 1966. — V. 14(3). — P. 493–498.
44. Damaschke P. Enumerating maximal bicliques in bipartite graphs with favorable degree sequences / P. Damaschke // Inf. Process. Lett. — 2014. — V. 114, Issue 6. — P. 317–321.
45. Delling D. Landmark-based Routing in Dynamic Graphs / D. Delling, D. Wagner // In Proc. International Workshop on Experimental Algorithms (WEA). Springer. — 2007. — 4525 of LNCS. — P. 52–65.
46. Delling D. Time-Dependent SHARC-Routing / D. Delling // J. Algorithmica. — 2011. — V. 60. — №. 1. — P. 60–94.
47. Di Puglia Pugliese L. A survey of resource constrained shortest path-problems: exact solution approaches / L. Di Puglia Pugliese, F. Guerriero // Networks. — 2013. — P. 183–200.
48. DIMACS Implementation Challenge. Shortest Paths [Электронный ресурс] : база данных дорожных графов. — 2006. — Режим доступа: http://www.dis.uniroma1.it/challenge9/.
49. Dreyfus S. An Appraisal of Some Shortest-Path Algorithms / S. Dreyfus // Operations Research. — 1969. — V. 17. — P. 395–412.

50. Dror M. Note on the complexity of the shortest path models for column generation in VRPTW / M. Dror // Operation Reseatch. — 1994. — V. 42. — P. 977–978.
51. Dumitrescu I. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem / I. Dumitrescu, N. Boland // Networks. — 2003. — V 42. — P. 135–153.
52. Eppstein D. Arboricity and bipartite subgraph listing algorithms / D. Eppstein // Information Processing Letters. — V. 5, Issue 4. — 1994. — P. 207–211.
53. Eppstein D. Finding the k shortest paths / D. Eppstein // Proceedings 35th Annual Symposium on Foundations of Computer Science. — 1994. — P. 154–165.
54. Faure N. Biclique completion problems for multicast network design / N. Faure, P. Chretienne, E. Gourdin, F. Sourd // Discrete Optimization. — 2007. — V. 4, Issues 3–4. — P. 360–377.
55. Ganter B. Formal concept analysis / B. Ganter, R. Wille // Springer, 1999. — 285 p.
56. Google Maps [Электронный ресурс] : геоинформационная система. — 2021. — Режим доступа: https://www.google.com/maps/
57. Goldberg A. Better landmarks within reach / A. Goldberg, H. Kaplan, R. Werneck // In Proc. International Workshop on Experimental Algorithms (WEA). Springer. — 2007. — 4525 of LNCS. — P. 38–51.
58. Goldberg A. Reach for A*: Efficient point-to-point shortest path algorithms / A. Goldberg, H. Kaplan, R. Werneck // Technical Report MSR-TR-2005-132, Microsoft Research. Microsoft Corporation. — 2005.
59. Graham R. L. On the addressing problem for loop switching / R. L. Graham, H. O. Pollak // The Bell System Technical Journal. — 1971. — V. 50. — №8. — P. 2495–2519.
60. Handler G. A Dual Algorithm for the Constrained Shortest Path Problem / G. Handler, I. Zang // Networks. — 1980. — V. 10. — P. 293–310.
61. Hassin R. Approximation Schemes for the Restricted Shortest Path Problem / R. Hassin // Mathematics Oper. Res. — 1992. — V. 17. — P. 36–42.

62. Hermelin D. Efficient enumeration of maximal induced bicliques / D. Hermelin, G. Manoussakis // Discrete Applied Mathematics. — 2020.
63. Horvath M. Solving resource constrained shortest path problems with LP-based methods / M. Horvath, T. Kis. // Computers. — 2016. — V. 73. — P. 150–164.
64. IBM Corp.: IBM ILOG CPLEX Optimizer Studio. CPLEX User's Manual. — Version 12 Release 6. — 2015.
65. Jepsen M, A branch-and-cut algorithm for the capacitated profitable tour problem / M. Jepsen, B. Petersen, S. Spoorendonk, D. Pisinger // Discrete Optimization. — 2014 — V. 14. — P. 78–96.
66. Joksch H. C. The Shortest Route Problem with Constraints / H. C. Joksch // Mathematical Analysis and Applecation. — 1966. — V. 14. — P. 191–197.
67. Jukna S. On covering graphs by complete bipartite subgraphs / S. Jukna, A. Kulikov // Discrete Mathematics. — 2009. — V. 309. — P. 3399–3403.
68. Kaufman D. E. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application / D. E. Kaufman, R. L. Smith // J. Intelligent Transportation Systems. — 1993. — V. 1. — №. 1. — P. 1–11.
69. Konstantinova E. V. Application of hypergraph theory in chemistry / E. V. Konstantinova, V. A. Skorobogatov // Discrete Mathematics. — 2001. — V. 235, Issues 1–3. — P. 365–383.
70. Kuznetsov S. Concept Stability for Constructing Taxonomies of Web-site users / S. O. Kuznetsov, D. I. Ignatov In Proc. Social Network Analysis and Conceptual Structures: Exploring Opportunities, Clermont-Ferrand — 2007.
71. Kuznetsov S. Comparing performance of algorithms for generating concept lattices / S. Kuznetsov, S. Obiedkov // J. Exp. Theor. Artif. Intell. — 2002. — V. 14. — P. 189–216.
72. Kuznetsov S. O. On Computing the Size of a Lattice and Related Decision Problems / S. O. Kuznetsov // Order. — 2001. — V. 18. — №. 4. — P. 313–321.
73. Li J. Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms / J. Li, G. Liu, H. Li, L. Wong // IEEE Transactions on Knowledge and Data Engineering. — 2007. — V. 19. — №12. — P. 1625–1637.

74. Lyu B. Maximum biclique search at billion scale /B. Lyu, L. Qin, X. Lin, Y. Zhang, Z. Qian, J. Zhou // Proc. VLDB Endow. — 2020. — P. 1359–1372.
75. Mehlhorn K. Resource constrained shortest paths / K. Mehlhorn, M. Ziegelmann // In Algorithms ESA. — 2000. — 1879 of LNCS.
76. Mongush Ch. M. On decomposition of a binary context without losing formal concepts / Ch. M. Mongush, V. V. Bykova // Journal of Siberian Federal University. Mathematics & Physics. — 2019. — № 3(12). — P. 323–330.
77. Nejad M. M. Hierarchical time-dependent shortest path algorithms for vehicle routing under ITS / M. M. Nejad, L. Mashayekhy, R. B. Chinnam, A. Phillips // J. IIE Transactions. — 2016. — V. 48. — №. 2. — P. 158–169.
78. Pathan A. K. Simulation Technologies in Networking and Communications: Selecting the Best Tool for the Test. / A. K. Pathan, M. M. Monowar, S. Khan // CRC Press. — 2017. — 648 p.
79. Prisner E. Bicliques in graphs I: bounds on their number / E. Prisner // Combinatorica. — 2000. — V. 20. — P. 109–117.
80. Shaham E. On finding the maximum edge biclique in a bipartite graph: a subspace clustering approach / E. Shaham, H. Yu, X. Li // Proc. of the 2016 SIAM International Conference on Data Mining (SDM). — 2016. — P. 315–323.
81. Sherali H. D. The time-dependent shortest pair of disjoint paths problem: Complexity, models, and algorithms / H. D. Sherali, K. Ozbay, S. Subramanian // J. Networks. — 1998. — V. 31. — №. 4. — P. 259— 272.
82. Soldatenko A. Algorithm for Analysis of Road Networks Described as Graphs and Hypergraphs / A. Soldatenko, D. Semenova // 2021 17th International Asian School-Seminar Optimization Problems of Complex Systems (OPCS). — 2021. — P. 121–125.
83. Soldatenko A. A. On accuracy of approximation for the resource constrained shortest path problem / A. A. Soldatenko // Journal of Siberian Federal University. Mathematics & Physics. — 2019. — № 12 (5). — P. 621–627.
84. Soldatenko A. A. On problem of finding all maximal induced bicliques of hypergraph / A. A. Soldatenko, D. V. Semenova // Journal of Siberian Federal University. Mathematics & Physics. — 2021. — № 14 (5). — P. 638–646.

85. Soldatenko A. A. Optimal Routing in Multi-Service Networks / A. A. Soldatenko, V. V. Bykova // 2018 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon). — 2018. — P. 1–4.
86. Van Mieghem P. Quality of Service Routing / P. Van Mieghem, F. A. Kuipers, T. Korkmaz, M. Krunz, M. Curado, E. Monteiro, X. Masip-BruinJ. Solé-ParetaS. Sánchez-López // In Quality of Future Internet Services. — 2003. — 2856 of LNCS. — P. 80–117.
87. Wagner D. Speed-up techniques for shortest-path computations / D. Wagner, T. Willhalm // In Proc. STACS. Springer. — 2007. — 4393 of LNCS. — P. 23–36.
88. Xue G. Polynomial Time Approximation Algorithms for Multi-Constrained QoS Routing / G. Xue, W. Zhang, J. Tang, K. Thulasiraman // IEEE/ACM Transactions on Networking. — 2008. — V. 16. — №3. — P. 656–669.
89. Zhang Y. On finding bicliques in bipartite graphs: a novel algorithm and its application to the integration of diverse biological data types / Y. Zhang, C. A. Phillips, G. L. Rogers, E. J. Baker, E. J. Chesler, M. A. Langston // BMC Bioinformatics. — 2014. — V. 15.
90. Zhong-Ji F. Efficient algorithm for extreme maximal biclique mining in cognitive frequency decision making / F. Zhong-Ji, L. Ming-Xue, H. Xiao-Xin, H Xiao-Hui, Z. Xin // IEEE 3rd International Conference on Communication Software and Networks. — 2011. — P. 25–30.
91. Zhu X. A three-stage approach for the resource-constrained shortest path as a sub-problem in column generation / X. Zhu, W. E. Wilhelm // Computers & Operations Research. — 2012. — V. 39. — Iss. 2. — P. 164–178.
92. Zverovich I. Bipartite bihypergraphs: A survey and new results / I. Zverovich, I. Zverovich // Discrete Mathematics. — 2006. — V. 306. — P. 801–811.

Список таблиц

Список иллюстраций

ПРИЛОЖЕНИЕ А. Акт о внедрении результатов в учебный процесс СФУ

СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ | SIBERIAN FEDERAL UNIVERSITY

МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Сибирский федеральный университет»
660041, Красноярский край,
г. Красноярск, проспект Свободный, д. 79
телефон: (391) 244-82-13, тел./факс: (391) 244-85-25
http://www.sfu-kras.ru, e-mail: office@sfu-kras.ru
ОКПО 02067876; ОГРН 1022402137460;
ИНН/КПП 2463011853/246301001

УТВЕРЖДАЮ

Проректор по учебной работе
ФГАОУ ВО «Сибирский федеральный университет»
Денис Сергеевич Гуц
«__» декабря 2021 г.

_____ № _____
на № _____ от _____

АКТ
о внедрении в учебный процесс результатов диссертационной работы на соискание учёной степени кандидата физико-математических наук Солдатенко Александра Александровича на тему «Разработка алгоритмов комбинаторной оптимизации для анализа графовых и гиперграфовых сетей»

Программы для ЭВМ «Программа HFindMCS выделения максимально полных подматриц (0,1)-матрицы», «Программа HFindMIB выделения максимальных индуцированных биклик гиперграфа» разработанные Солдатенко Александром Александровичем, и теоретические результаты кандидатской диссертации «Разработка алгоритмов комбинаторной оптимизации для анализа графовых и гиперграфовых сетей», выполненной в Сибирском федеральном университете, внедрены в учебный процесс на кафедре Высшей и прикладной математики. Эти материалы используются при подготовке бакалавров по специальностям 01.03.01 – Математика, 01.03.02 – Прикладная математика и информатика, 02.03.01 – Математика и компьютерные науки, при изучении дисциплин «Комбинаторные алгоритмы», «Технологии хранения и обработки больших данных», а также в НИР магистрантов по направлению подготовки 01.04.02.06 – Прикладная математика и информатика в гуманитарных и социально-экономических науках.

Директор Института математики и фундаментальной информатики СФУ
канд. физ.-мат. наук, доцент _____ О.Н. Черепанова

Председатель учебно-методического совета Института математики и фундаментальной информатики СФУ
д-р физ.-мат. наук, профессор _____ Е.К. Лейнартас

Заведующий кафедрой Высшей и прикладной математики
д-р физ.-мат. наук, профессор _____ С.Г. Мысливец

ПРИЛОЖЕНИЕ Б. Акт об использовании результатов в производственно-технологическом процессе ООО «Ар Ди Сайнс»

УТВЕРЖДАЮ
Руководитель группы аналитики
ООО «Ар Ди Сайнс»
канд.техн. наук Сергеева Наталья Александровна
«30» сентября 2021 г.

АКТ

об использовании в производственно-технологическом процессе ООО «Ар Ди Сайнс» результатов диссертационной работы на соискание ученой степени канд. физ.-мат. наук по специальности 05.13.17 – Теоретические основы информатики

Солдатенко Александра Александровича

на тему «Разработка алгоритмов комбинаторной оптимизации для анализа графовых и гиперграфовых сетей»

В диссертационной работе Солдатенко А.А. предложены алгоритмы комбинаторной оптимизации для решения задач о кратчайшем пути с ограничениями и задач перечислительного типа применительно к гиперграфам. Актуальность проведения исследований обусловлена тем, что реальные системы и приложения обычно предъявляют дополнительные требования к кратчайшим путям, например, ограничение по доступным ресурсам или зависимость от времени, а гиперграфы представляют мощный инструмент для моделирования сетей (дорожных, телекоммуникационных, семантических). Алгоритмы, разработанные в диссертации, ориентированы на решение именно таких задач.

Результаты диссертационной работы Солдатенко А.А., в частности, программы для ЭВМ «Модифицированный алгоритм ALT», «Программа RevTree для поиска ресурсоограниченного пути в сети» возможно интегрировать в программное обеспечение «Аналитическая транспортная система R.D.Science OPENROAD Platform» (регистрационный номер в Реестре программ для ЭВМ №2020615899 от 03.06.2020), и «Программа HFindMIB выделения всех максимальных индуцированных биклик» возможно использовать для анализа семантических сетей, представленных в виде двудольного графа, где вершины одной доли соответствуют объектам, а другой доли – отношениям.

Использование указанных результатов при интеллектуальном анализе дорожного графа города Красноярска в условиях ограничений позволяет повысить качество работы алгоритмов выбора стратегий управления и дальнейшей оптимизации планов координаций для светофорных объектов участка дорожной сети.

Ведущий специалист
канд. техн. наук, доцент /Чубарова О.В.

Программист-разработчик
канд. техн. наук /Кононова Н.В.