Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(национальный исследовательский университет)

На правах рукописи

ШАНГИН Роман Эдуардович

РАЗРАБОТКА И АНАЛИЗ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ РАЗМЕЩЕНИЯ ГРАФА

Специальность 05.13.17 – теоретические основы информатики

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель
ПАНЮКОВ Анатолий Васильевич
доктор физ.-мат. наук, профессор

Челябинск – 2015

Оглавление

Введение

Актуальность темы

Анализ и решение задач оптимального размещения объектов является одним из интенсивно развивающихся направлений современной дискретной оптимизации. Задачи такого класса имеют важное прикладное значение и возникают в различных сферах деятельности: при проектировании и оптимизации логистической системы предприятий, при управлении вычислительными ресурсами в распределенных гетерогенных вычислительных средах, при расчете оптимального расположения технологического оборудования в цехах и т.д.

Степень разработанности темы

Класс задач размещения интересен как с практической, так и с теоретической точки зрения, о чем свидетельствует большое количество работ, посвященных данной проблематике. Среди них в первую очередь стоит отметить работы Забудского Г.Г., Кочетова Ю.А., Панюкова А.В., Еремеева А.В., Гимади Э.Х., Береснева В.Л., Колоколова А.А., Васильева И.Л., Дементьева В.Т., Гермейера Ю.Б., Шамардина Ю.В., Агеева А.А., Антипина А.С., Хамисова О.В. и др. На сегодняшний день область дискретной оптимизации, связанная с задачами размещения, активно развивается. Ведутся исследования новых постановок задач, развиваются точные и приближенные методы их решения, выделяются полиномиально разрешимые случаи.

Можно выделить два вида задач оптимального размещения: задачи размещения взаимосвязанных объектов и задачи размещения-распределения. Отличие состоит в том, что в задачах первого класса необходимо найти места расположения объектов, между которыми имеются некоторые связи (не обязатель-

но между всеми). В задачах второго класса связи устанавливаются в результате их решения. Например, при размещении пунктов обслуживания (логистические центры, телекоммуникационные узлы) к ним прикрепляются клиенты (торговые точки, пользователи услуг связи), в результате чего устанавливаются связи.

Классическими представителями первого класса являются задача Вебера и квадратичная задача о назначениях. Разработкой этой проблематики занимались Ахмедов И.С., Демиденко В.А., Жак С.Б., Зинченко А.Б, Иорданский М.А., Панюков А.В., Сергеев С.И., Сигал И.Х., Стоян Ю.Г., Трубин В.А., Яковлев С.В., Adolfson D., Beckmann M.J., Burcard R.E., Francis R.L., Koopmans T.C., Tamir A., Wesolowsky G.O. и другие. Наиболее известные задачи второго класса: простейшая задача размещения, задачи о p-медиане и о p-центре. Заметный вклад в их исследование внесли Агеев А.А., Бахтин А.Е., Береснев В.Л., Гимади Э.Х., Глебов Н.И., Дементьев В.Т., Емеличев В.А., Колоколов А.А., Кочетов Ю.А., Леванова Т.В., Лореш М.А., Hakimi S.L., Kariv O., Krarup J., Pruzan P.M. и ряд других авторов.

Основные критерии, по которым ведется классификация рассматриваемого класса задач следующие:

- трактовка понятия объект размещения и его геометрическая форма;
- структура связей между размещаемыми объектами;
- критерии оценки качества решений;
- структура области размещения и другие.

Понятие объект трактуется достаточно широко. Например, при размещении вычислительных задач в узлах распределенной вычислительной среды, в качестве объекта выступает отдельное вычислительное задание. При проектировании генеральных планов предприятий под объектом понимаются здания и сооружения; при создании робототехнологических комплексов – единицы тех-

нологического оборудования. Если проектируется печатная плата, то объектами являются элементы печатного монтажа, к примеру, различные микросхемы. При расположении пунктов обслуживания объектами выступают станции скорой или технической помощи, магазины, склады и т.д. Также важным параметром задачи является *геометрическая форма размещаемого объекта*. В зависимости от ситуации размещаемые объекты можно рассматривать как материальные точки, либо необходимо учитывать их габариты, которые также являются одной из характеристик задач размещения. Часто протяженные плоские объекты, например, единицы технологического оборудования, аппроксимируются прямоугольниками.

Что касается *типа связи* между объектами, то он зависит от сферы деятельности, в которой возникает задача размещения. Например, в размещении вычислительных задач – это потоки данных, передаваемые между отдельными задачами. При проектировании нефтехимического предприятия, его объекты взаимосвязаны различными трубопроводами. При проектировании печатных плат связи – это проводники, соединяющие элементы печатного монтажа, а когда размещаются пункты обслуживания, связями являются, например, потоки товаров.

При решении задач оптимального размещения необходимо учитывать различные *ограничения на расположение* объектов. Строительные правила и нормы определяют санитарные, противопожарные и другие нормативы, которые при проектировании генпланов задают ограничения на минимально допустимые расстояния между сооружениями и единицами оборудования. Между объектами максимально допустимые расстояния могут определяться технологией производства, к примеру, мощностью насосных станций. А при создании атомных станций создается «зеленая» зона, в которой запрещается размещение населенных пунктов. Также во многих случаях необходимо «регулярное» расположение объектов, например, установление оборудования вдоль «красных»

линий, что позволяет выделять кварталы и проектировать прямые проезды в расположении объектов.

В зависимости от того какие цели ставятся перед нами, рассматриваются различные *критерии оценки* качества получаемых размещений. Например, для нефтехимических предприятий стоимость трубопроводных связей может составлять около 25 процентов от общих капитальных затрат на строительство. Поэтому при размещении оборудования такого предприятия на заданной строительной площадке можно минимизировать суммарную стоимость трубопроводных связей. Довольно часто используются критерии минимума площади, которой занимают размещенные объекты, и минимума наиболее дорогой по стоимости (максимальной) связи. При проектировании печатных плат критерии размещения нацелены, в основном, на облегчение выполнения последующей трассировки. Это может быть, к примеру, минимум числа пересечений проводников. В последнее время изучаются задачи размещения, в которых расстояние от размещаемого объекта до ближайшего фиксированного должно быть максимально возможным. Такие задачи появляются, например, при размещении опасных (вредных) производств, либо, наоборот, требующих экологической чистоты.

Одной из основных характеристик задач оптимального размещения объектов является *структура области*, в которой производится размещение, к примеру, она может быть непрерывной или дискретной. Размерность непрерывной области может быть различной: одномерной линия, двумерной плоскость и т.д. Существенное значение для анализа и разработки методов решения подобных задач имеет метрика, которая занимается измерением расстояния между объектами. Причем в одной модели могут использоваться различные метрики. Например, в евклидовой метрике измеряются минимально допустимые расстояния между объектами, а в прямоугольной метрике – длина связей . В дискретных моделях указывается конечное число мест, то есть позиций, для установки объ-

ектов, которые могут находиться, к примеру, на плоскости либо на некоторой сети.

Различный математический аппарат используется при описании оптимизационных моделей задач размещения, в частности, модели и методы вычислительной геометрии, комбинаторного анализа, теории графов, линейного и целочисленного программирования. Методы решения задач оптимального размещения определяются характеристиками используемой модели. В основном, такие задачи являются $\mathcal{NP}$-трудными, но для отдельных классов задач известны полиномиальные алгоритмы.

Важным классом задач оптимального размещения является класс *нелинейных задач о назначениях* [117]. Настоящая работа посвящена одной из наиболее значимых задач из данного класса – *дискретной задаче Вебера* (далее ДЗВ) [8,9,15,16,23]. Дискретная задача Вебера представляет собой ослабленный вариант хорошо известной *квадратичной задачи о назначениях* [10,11,21,22], в котором требование инъективности и сюръективности отображения множества вершин графа в множество позиций размещения снимается. Иными словами, в ДЗВ требуется найти оптимальное отображение (размещение) вершин графа в конечном множестве позиций с целью минимизации некоторого критерия, когда допускается размещение нескольких вершин графа в одной позиции, а также случай, когда ни одной вершины в позиции не размещено. Поскольку в общей постановке ДЗВ является NP-трудной, то представляются перспективными ее исследования в следующих направлениях: выделение подклассов задачи, для которых возможно построение точных полиномиальных алгоритмов, построение приближенных алгоритмов с гарантированными оценками точности, построение эффективных эвристик и метаэвристик.

Цель и задачи исследования

Целью диссертации является разработка комбинаторных алгоритмов решения дискретной задачи Вебера.

Для достижения выбранной цели необходимо решить следующие задачи.

1. Исследовать свойства дискретной задачи Вебера в графовой постановке.
2. Найти новые подклассы задачи, разрешимые точно за полиномиальное время.
3. Найти новые подклассы задачи, разрешимые с гарантированной оценкой точности за полиномиальное время.
4. Разработать эффективные алгоритмы точного и приближенного решения найденных новых подклассов задачи.
5. Реализовать предложенные алгоритмы в виде компьютерных программ, и проверить их эффективность на числовых примерах.

Научная новизна

В работе получены новые теоретические результаты и разработаны новые алгоритмы решения дискретной задачи Вебера. Среди полученных результатов можно выделить следующие.

1. Доказано, что ДЗВ полиномиально разрешима при фиксированном k, когда размещаемый граф представляет собой k-дерево либо k-цепь. Разработаны новые эффективные алгоритмы точного решения ДЗВ на данных классах графов.
2. Разработан новый полиномиальный приближенный алгоритм решения ДЗВ. Доказана его апостериорная оценка точности для случая произвольного графа. Доказаны гарантированные априорные и асимптотические оценки точности предложенного алгоритма для некоторых подклассов задачи.

3. Построена эффективная метаэвристика с апостериорной оценкой точности решения ДЗВ для случая произвольного графа, сочетающая идеи предложенного приближенного алгоритма и генетического алгоритма.

Теоретическая и практическая значимость работы

Теоретическая значимость работы заключается в нахождении новых подклассов дискретной задачи Вебера, которые могут быть разрешены точно либо с гарантированной оценкой погрешности за полиномиальное время, а также в разработке и анализе новых эффективных алгоритмов поиска точных и приближенных решений исследуемой задачи.

Практическая значимость работы работы состоит в том, что ее теоретические результаты могут служить основой для разработки программно-алгоритмического обеспечения решения реальных прикладных задач в области проектирования и оптимизации функционирования логистических систем предприятий, а также при решении задач управления вычислительными ресурсами в распределенных гетерогенных вычислительных средах. Разработанные алгоритмы реализованы в зарегистрированном программном обеспечении, созданном в рамках Госконтракта № 11426р/17183. Алгоритмы показали свою эффективность и могут применяться как при решении практических задач, так и в рамках преподавания курсов «Исследование операций» и «Теория принятия решений».

Методология и методы исследования

В диссертации использовались методы дискретного и комбинаторного анализа, методы исследования операций, элементы теории графов, теория вычислительной сложности, методы локального поиска, теория построения приближенных алгоритмовтеория построения алгоритмов поиска приближенных решений.

Положения, выносимые на защиту

1. Сформулирована и доказана теорема о том, что дискретная задача Вебера полиномиально разрешима при фиксированном k на классе k-деревьев. Предложен точный алгоритм решения, основанный на технике динамического программирования по дереву декомпозиции, имеющий оценку времени счета $O(n^{k+3})$ и оценку требуемой оперативной памяти $O(n^{k+3})$, где n – число вершин графа.

2. Сформулирована и доказана теорема о том, что дискретная задача Вебера полиномиально разрешима при фиксированном k на классе k-цепей. Предложен точный алгоритм решения, основанный на технике динамического программирования, имеющий оценку времени счета $O(n^{k+2})$ и оценку требуемой оперативной памяти $O(n^k)$, где n – число вершин графа.

3. Предложен приближенный алгоритм решения дискретной задачи Вебера в случае произвольного размещаемого графа, имеющий трудоемкость $O(n^3)$. Доказана его апостериорная оценка точности и выявлены параметры задачи, влияющие на величину данной оценки. Найдены подклассы задачи, на которых алгоритм является 2-приближенным и асимптотически точным.

4. Для дискретной задачи Вебера с произвольным размещаемым графом предложена метаэвристика, позволяющая вычислять апостериорную оценку погрешности найденного решения. Метаэвристика находит приближенные решения близкие к оптимальным и превосходит в точности решения наилучшую из известных метаэвристик, при сравнимом времени счета.

Степень достоверности результатов

Достоверность полученных результатов обеспечивается строгостью математических постановок и доказательств утверждений, корректным использованием методов дискретного и комбинаторного анализа, подтверждением теоретических результатов численными экспериментами. Результаты работы основываются на известных достижениях теории графов, теории принятия решений, экономической кибернетики, теории информации.

Апробация результатов исследования

Основные положения диссертационной работы, разработанные алгоритмы и результаты вычислительных экспериментов прошли апробацию на следующих международных и всероссийских научных конференциях:

– Всероссийская конференция по математическим методам и статистике, Москва, 2011.

– Всероссийская конференция «Статистика. Моделирование. Оптимизация», Челябинск, 2011.

– V Всероссийская конференция «Проблемы оптимизации и экономические приложения», Омск, 2012.

– XIII Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям, Новосибирск, 2012.

– XIV Всероссийский Симпозиум по прикладной и промышленной математике, Сочи-Вардане, 2012.

– Международная конференция «Дискретная оптимизация и исследование операций», Новосибирск, 2013.

– Международная конференция «Информационные технологии интеллектуальной поддержки принятия решений», Уфа, 2013.

– XII Всероссийская конференция «Компьютерная безопасность и криптография», Томск, 2013.

– Международная конференция «The 15th International Workshop on Computer Science and Information Technologies», Уфа, 2013.

– II Международная конференция «Высокопроизводительные вычисления – математические модели и алгоритмы», Калининград, 2013.

– 45-ая Международная школы-конференция, посвященная 75-летию В.И. Бердышева, Екатеринбург, 2014.

– XIII Всероссийская конференция «Компьютерная безопасность и криптография», Томск, 2014.

– Научный семинар в «Лаборатория алгоритмов и технологий анализа сетевых структур» (ЛАТАС), Нижний Новгород, 2014.

– Международная конференция «The Second International Conference on Information Technology and Quantitative Management», Москва, 2014.

– XII Всероссийское совещание по проблемам управления, Москва, 2014.

– XVI Международная школа-семинар «Методы оптимизации и их приложения», Иркутск-Ольхон, 2014.

– Научный семинар «Дискретные экстремальные задачи», Новосибирск, 2015.

– XV Всероссийская конференция «Математическое программирование и приложения», Екатеринбург, 2015.

Основные результаты диссертации опубликованы в 6 научных статьях в журналах, входящих в перечень ВАК, в 10 статьях в различных журналах,

сборниках и материалах конференций. Общее число публикаций — 16. Зарегистрирована одна программа для ЭВМ.

Структура и объем диссертации

Диссертация состоит из введения, обзора литературы, четырех глав, заключения и списка литературы (141 наименование). Объем диссертации – 116 страниц.

Содержание работы

В первой главе, «Об исследуемой задаче», приводится обзор известных в литературе постановок задачи Вебера. Рассматривается класс нелинейных задач о назначениях. Дается формулировка дискретной задачи Вебера в терминах отображений и целочисленного линейного программирования. Показывается связь исследуемой дискретной задачи Вебера с классической непрерывной задачей Вебера и известной квадратичной задачей о назначениях. Приводится обзор известных в литературе результатов, полученных для исследуемой задачи. Дается обзор направлений практического использования задачи.

Во второй главе, «Точные алгоритмы решения задачи для графов специального вида», обсуждаются новые подклассы задачи ДЗВ, которые могут быть разрешены за полиномиальное время.

Для решения подкласса задач, когда размещаемый граф является n-вершинным k-деревом, предложен точный алгоритм A_T, полиномиальный при фиксированном k. Предложеный алгоритм обобщает известный алгоритм решения ДЗВ для последовательно-параллельного графа, предложенный Ф. Малучелли. Доказана теорема о том, что такой алгоритм всегда находит точное решение ДЗВ, когда размещаемый граф есть k-дерево. Определена трудоемкость алгоритма – $O(n^{k+3})$ и оценка требуемой алгоритмом памяти – $O(n^{k+3})$. Предложенный алгоритм A_T обобщен на класс графов с ограниченным параметром древовидной ширины. Доказано, что если параметр древовидной ширины размещаемого графа равен k, то такой случай ДЗВ

также может быть разрешен за полиомиальное время алгоритмом, имеющим трудоемкость $O(n^{k+3})$ и оценку памяти $O(n^{k+3})$.

Для решения подкласса задач, когда размещаемый граф является n-вершинной k-цепью, предложен точный полиномиальный при фиксированном k алгоритм A_C, основанный на динамическом программировании. Доказана теорема о том, что алгоритм A_C всегда находит точное решение ДЗВ, когда размещаемый граф есть k-цепь. Оценка времени работы алгоритма – $O(n^{k+2})$, оценка требуемого объема оперативной памяти – $O(n^k)$, где n – число вершин графа. Благодаря учету специфики задачи, данный алгоритм имеет лучшие оценки времени работы и объема требуемой оперативной памяти, чем более общий алгоритм для k-дерева. Предложенный алгоритм A_C обобщен на класс графов с ограниченным параметром ленточной ширины. Доказано, что если параметр ленточной ширины размещаемого графа равен k, то такой случай ДЗВ также может быть разрешен за полиомиальное время алгоритмом, имеющим трудоемкость $O(n^{k+2})$ и оценку памяти $O(n^k)$.

Для частного случая ДЗВ, когда размещаемый граф является простым циклом, предложен точный алгоритм A_S, основанный на динамическом программировании. Доказана теорема о том, что алгоритм A_S всегда находит точное решение ДЗВ, когда размещаемый граф есть простой цикл. Определена трудоемкость алгоритма – $O(n^4)$ и оценка требуемой алгоритмом памяти – $O(n)$. Благодаря учету специфики задачи, данный алгоритм A_S требует значительно меньше оперативной памяти, чем более общий алгоритм Ф. Малучелли для последовательно-параллельных графов, хотя оценки трудоемкостей таких алгоритмов одинаковы.

Третья глава, «Алгоритмы с оценками точности», посвящена алгоритмам с оценками точности решения ДЗВ. Для частного случая ДЗВ, когда размещаемый граф является простым циклом, предложен алгоритм прибли-

женного решения с трудоемкостью $O(n^3)$, частично использующий идею точного алгоритма A_C решения ДЗВ для k-цепи. Доказано, что данный алгоритм является 2-приближенным и асимптотически точным.

Для ДЗВ с произвольным размещаемым графом предложен приближенный алгоритм Apx решения с апостериорной оценкой точности, использующий в своей работе известный точный алгоритм решения ДЗВ для корневого дерева. Трудоемкость алгоритма равна $O(n^3)$. Исследовано влияние различных параметров задачи на величину оценки. Найден подкласс задач, на котором такой алгоритм является 2-приближенным. Определены подклассы задачи, на которых использование даного приближенного алгоритма перспективно.

Для ДЗВ с произвольным размещаемым графом предложена метаэвристика ApxGA, позволяющая вычислять апостериорную оценку погрешности найденного решения. Данная метаэвристика сочетает в себе идеи предложенного приближенного алгоритма Apx и генетического алгоритма. Алгоритм Apx используется в метаэвристике как для рассчета начальной популяции, так и для решения задачи оптимального кроссиговера на каждой итерации генетического алгоритма.

В четвертой главе, «Вычислительные эксперименты», представлены результаты вычислительных экспериментов по анализу эффективности предложенных в диссертационной работе комбинаторных алгоритмов.

Приведены результаты по анализу времени работы точных алгоритмов, предложенных в главе 2, в сравнении с коммерческим пакетом IBM ILOG CPLEX. Было выявлено, что применение алгоритмов A_C и A_T для решения ДЗВ перспективно при сравнительно малых значениях величины параметра k (от 1 до 3), причем, чем меньше величина k и чем больше количество вершин размещаемого графа, тем предложенные алгоритмы более эффективны по сравнению с алгоритмом ветвей и границ, реализованном в пакете IBM ILOG CPLEX.

Приведены результаты анализа эффективности приближенных алгорит-

мов, предложенных в главе 3, как в сравнении с коммерческим пакетом IBM ILOG CPLEX, так и в сравнении с известной метаэвристикой поиска с чередующимися окрестностями, предложенной В. Филиповичем и Д. Кратика [97]. Выявлены параметры задачи, влияющие на величину оценки точности предложенных приближеных алгоритмов. Найдены подклассы задачи, на которых использование приближенного алгоритма Apx перспективно. Обнаружено, что предложенная метаэвристика ApxGA позволяет получать приближенные решения близкие к оптимальным и существенно превосходит известную метаэвристику В. Филиповича и Д. Кратика в точности вычисляемого решения, при сравнимом времени счета.

В заключении подведены основные итоги данной работы, сформулированы результаты, представляемые диссертантом к защите, а также предложены некоторые перспективные направления дальнейших исследований дискретной задачи Вебера.

В приложении приводятся основные обозначения, используемые в диссертационной работе.

Глава 1.

Об исследуемой задаче

В настоящей главе приводится обзор известных в литературе постановок непрерывной задачи Вебера. Рассматривается класс нелинейных задач о назначениях. Приводится постановка исследуемой в работе задачи – дискретной задачи Вебера. Дается формулировка исследуемой задачи в терминах отображений и целочисленного линейного программирования. Показывается связь исследуемой задачи с классической непрерывной задачей Вебера и с квадратичной задачей о назначениях. Приводится обзор известных в литературе результатов, полученных другими авторами для исследуемой задачи. Дается обзор направлений практического использования рассматриваемой задачи.

§1.1. Обзор постановок задачи Вебера

Первые исследования задач оптимального размещения взаимосвязанных объектов относятся к 17 столетию, когда П. Ферма сформулировал задачу, известную сейчас как задача Вебера: найти такую точку на плоскости, чтобы сумма расстояний от нее до трех фиксированных точек была минимальной. Задача была решена геометрически Э. Торричелли в 1640 году. В 1750 году Т. Симпсон обобщил задачу на случай, когда фиксированные объекты имеют веса. В 1909 году Г. Вебер использовал подобную модель для определения оптимального расположения фабрики при заданных точках размещения ресурсов [141]. Однако только с появлением и развитием математической кибернетики эти задачи стали предметом всестороннего и глубокого исследования [7].

В настоящее время различают два типа задачи Вебера. Первый тип задачи Вебера – *Multisource Weber Problem* (далее *MSW*) [46,68,89,103,106,141], когда требуется разместить несколько предприятий, производящих один и тот же продукт (либо оказывающих одну услугу), с целью минимизации суммарных транспортных затрат между клиентами и наиближайшими к ним размещенными предприятиями. Заметим, что задача первого типа принадлежит классу задач размещения-распределения. Второй тип задачи Вебера – *Multifacility Weber problem* (далее *MFW*) [53, 72, 73, 94, 95, 140], когда требуется разместить несколько взаимосвязанных предприятий, производящих разнородную продукцию (либо оказывающих различные услуги), с целью минимизации суммарных транспортных затрат как между всеми предприятиями, так и между предприятиями и клиентами. Задача второго типа принадлежит классу задач размещения взаимосвязанных объектов.

Приведем постановку задачи *MSW*, представляющей собой задачу Вебера первого типа. Дана плоскость $L = \{x : x \in \mathbb{R}^2\}$. Пусть p – число предприятий, которые необходимо разместить на плоскости L с целью обслуживания m клиентов, положение которых зафиксировано на плоскости L. Далее под предприятиями и клиентами будем понимать точечные объекты без габаритов. Пусть величина w_j – удельная стоимость снабжения клиента j. Обозначим $X_i = (x_i, y_i), X_i \in L$ координаты размещения предприятия $i, i = 1, ..., p$. Пусть $A_j = (a_j, b_j), A_j \in L$ – *заданные* координаты размещения клиента $j, j = 1, ..., m$. Пусть $d(X_i, A_j)$ расстояние между предприятием i, размещенным в X_i, и фиксированным клиентом j. Ограничения на производственные мощности предприятий не предполагаются. Пусть $z_{ij} \in \{0, 1\}$ булева переменная, где $z_{ij} = 1$, если клиент j прикреплен к предприятию i, иначе $z_{ij} = 0$.

Необходимо разместить p объектов на плоскости L таким образом, чтобы суммарная стоимость связей между каждым клиентом и ближайшим к нему размещенным предприятием была минимальной. Приведем известную форму-

лировку задачи в терминах частично целочисленного линейного программирования [115].

$$\sum_{i=1}^{p}\sum_{j=1}^{m} z_{ij} w_j d(X_i, A_j) \to \min_{X,z} \tag{1.1}$$

при ограничениях

$$\sum_{i=1}^{p} z_{ij} = 1, \qquad j = 1, ..., m; \tag{1.2}$$

$$z_{ij} \in \{0, 1\}, \qquad i = 1, ..., p;\ j = 1, ..., m; \tag{1.3}$$

$$X_i \in L, \qquad i = 1, ..., p. \tag{1.4}$$

Суммирование в целевой функции (1.1) осуществляется по всем парам «предприятие-клиент», причем если для некоторого предприятия i и некоторого клиента j переменная $z_{ij} = 1$, то стоимость связи (транспортные затраты на поставку товаров) между i и j равна величине $w_j d(X_i, A_j)$, иначе стоимость связи равна 0. Ограничение (1.2) имеет смысл того, что каждый клиент должен быть прикреплен к одному и только одному предприятию. Геометрическая интерпретация данной задачи представлена на рисунке 1.1.

Известно, что задача (1.1-1.4) NP-трудна даже для евклидовой и прямоугольной метрик [115]. Задача *MSW* была исследована множеством авторов. Для ее решения были разработаны точные алгоритмы, основанные на методе ветвей и границ [107], предложены эффективные эвристики и метаэвристики [68, 106, 141]. Важно заметить, что дискретная постановка данной задачи, когда требуется разместить предприятия не на плоскости L, а в *конечном множестве* точек $P, |P| = n$, где $P \subset L$, получила большую популярность и известна как задача о p-медиане [3, 85, 120].

Будем обозначать $X_i = (x_i, y_i), X_i \in L$ – координаты позиции $i, i = 1, ..., n$, в которой может быть размещено предприятие. Приведем математическую формулировку задачи о p-медиане в терминах целочисленного линейного програм-

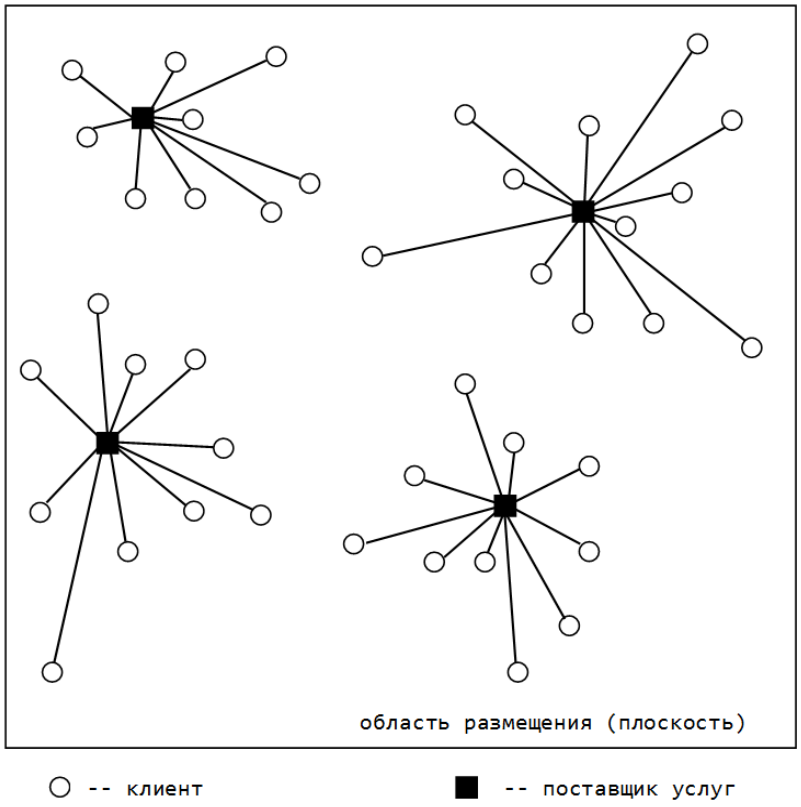

Рис. 1.1. Геометрическая интерпретации задачи *MSW*

мирования [82]. Булевы переменные в задаче будут иметь следующий смысл. Пусть $t_i \in \{0,1\}$ булева переменная, где $t_i = 1$, если в позицию $i \in P$ размещено некоторое предприятие, иначе $t_i = 0$. Пусть $z_{ij} \in \{0,1\}$ булева переменная, где $z_{ij} = 1$, если клиент j снабжается из предприятия, размещенного в позиции $i \in P$, иначе $z_{ij} = 0$.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij} w_j d(X_i, A_j) \to \min_{t,z} \tag{1.5}$$

при ограничениях

$$\sum_{i=1}^{n} z_{ij} = 1, \qquad j = 1, ..., m; \tag{1.6}$$

$$\sum_{t=1}^{n} t_i = p; \tag{1.7}$$

$$z_{ij} \le t_i, \qquad i = 1, ..., n;\ j = 1, ..., m; \tag{1.8}$$

$$t_i \in \{0,1\}, \qquad i = 1, ..., n; \tag{1.9}$$

$$z_{ij} \in \{0,1\}, \qquad i = 1, ..., n;\ j = 1, ..., m; \tag{1.10}$$

В целевой функции суммирование происходит по всем парам «позиция-клиент», причем если в некоторой позиции i размещено предприятие и клиент j прикреплен к нему, то стоимость связи между предприятием, размещенным в i, и клиентом j равна величине $w_j d(X_i, A_j)$, иначе стоимость связи равна 0. Ограничение (1.6) говорит о том, что каждый клиент j может быть прикреплен к одному и только одному предприятию. Ограничение (1.7) подразумевает, что сумма всех позиций, в которых размещены предприятия, равна p. Ограничение (1.8) говорит о том, клиент j может быть обслужен из позиции i, только в том случае, если в позицию i размещено предприятие.

Задача о p-медиане принадлежит классу NP-трудных в сильном смысле задач, поскольку к ней сводится задача о вершинном покрытии [64]. Задача о p-медиане возникает во многих приложениях, например, размещение предприятий бытового обслуживания, складов, пунктов автосервиса на дорогах, коммутаторов в телефонной сети и др. [108]. Данная задача была исследована множеством авторов. Обширный литературный обзор по этой задаче можно найти в [93], а также в [33]. Используя модель целочисленного линейного программирования задачи о p-медиане можно применять аппарат целочисленной оптимизации: алгоритмы ветвей и границ [118], ветвей и отсечений [4], перебора L-классов [13] и другие. Отдельно хочется отметить последние работы, касающиеся решения задачи о p-медиане большой размерности: VNS [83], GRASP эвристики [121], метод ветвей, отсечений и оценок [33].

Приведем постановку задачи *MFW*, которая представляет собой задачу Вебера второго типа. Дана плоскость $L = \{x : x \in \mathbb{R}^2\}$. Пусть $J = \{1, ..., m\}$ – множество номеров фиксированных объектов на плоскости L (клиенты) и $V = \{1, ..., n\}$ – множество номеров объектов, которые необходимо разместить

на той же плоскости (предприятия). Координаты фиксированного объекта с номером j обозначим через $A_j = (a_j, b_j) \in L$, координаты размещаемого объекта с номером i – через $X_i = (x_i, y_i) \in L$. Удельные стоимости связей между j-м фиксированным и i-м размещаемым объектами обозначим через u_{ij}, а через w_{ik} – удельные стоимости связей между размещаемыми объектами i и k, при этом считаем, что $w_{ik} = w_{ki}, k \neq i$. Пусть $d(X_i, A_j)$ – расстояние между j-м фиксированным и i-м размещаемым объектами, а $g(X_i, X_k)$ – расстояние между i-м и k-м размещаемыми объектами. Требуется разместить предприятия так, чтобы суммарная стоимость связей между всеми объектами была минимальной, то есть

$$\sum_{i \in V} \sum_{k \in V} w_{ik} g(X_i, X_k) + \sum_{i \in V} \sum_{j \in J} u_{ij} d(X_i, A_j) \to \min_X . \tag{1.11}$$

Для прямоугольной и евклидовой метрик задача записывается следующим образом:

$$\sum_{i \in V} \sum_{k \in V} w_{ik} (|x_i - x_k| + |y_i - y_k|) + \sum_{i \in V} \sum_{j \in J} u_{ij} (|x_i - a_j| + |y_i - b_j|) \to \min_{x,y}; \tag{1.12}$$

$$\sum_{i \in V} \sum_{k \in V} w_{ik} \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} + \sum_{i \in V} \sum_{j \in J} u_{ij} \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \to \min_{x,y} . \tag{1.13}$$

Задача (1.11), а также ее частные случаи (1.12) и (1.13) были исследованы множеством авторов [73, 74]. Доказано, что задачи (1.12) и (1.13) принадлежат классу задач выпуклого программирования, а значит могут быть решены точно за полиномиальное время [91, 140]. Геометрическая интерпретация данной задачи представлена на рисунке 1.2.

Несмотря на то, что такие задачи на плоскости полиномиально разрешимы, любопытно заметить следующий факт. Пусть дан граф H, вершины и ребра которого целиком принадлежат плоскости L. Обозначим $P(H), P(H) \subset L$ – множество точек плоскости, принадлежащих графу H. В работе [132] доказано,

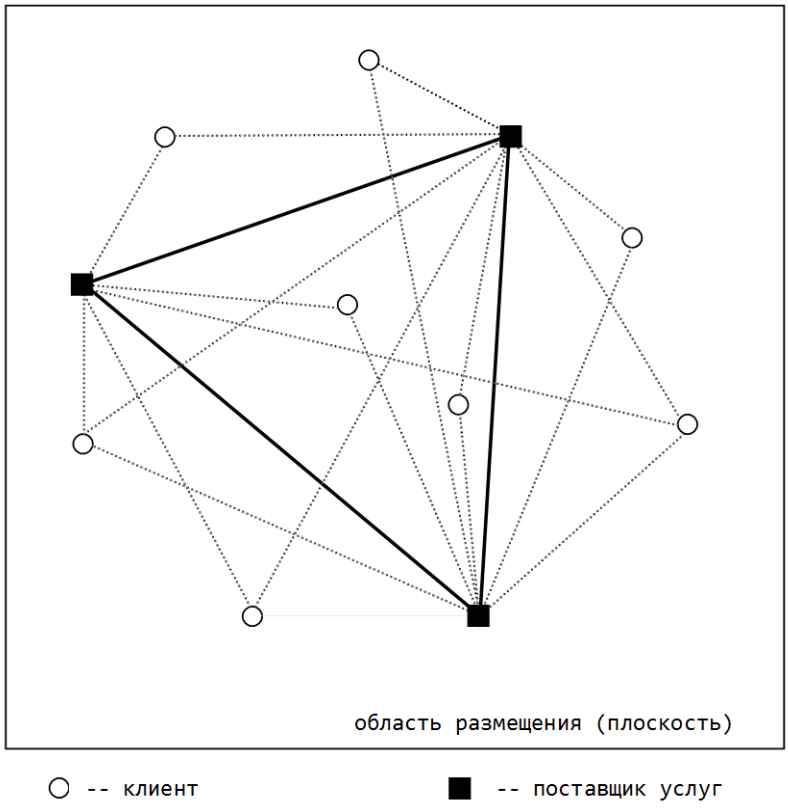

Рис. 1.2. Геометрическая интерпретации задачи *MFW*

что если область размещения предприятий ограничена лишь множеством $P(H)$ и H – произвольный граф, то данная задача становится NP-трудной. Задача с таким ограничением на область размещения известна в зарубежной литературе под названием *n-Facility Minisum Problem* [132]. Представляет большой интерес случай задачи, когда область размещения предприятий ограничена *только вершинами* графа H, то есть конечным множеством точек на плоскости L. Очевидно, такая задача также NP-трудна. Сформулируем такую задачу в терминах целочисленного линейного программирования.

Дана плоскость $L = \{x : x \in \mathbb{R}^2\}$. Пусть $J, |J| = m$ – множество клиентов (точек), фиксированных на плоскости L, где $A_j = (a_j, b_j) \in L$ – координаты объекта $j \in J$. Обозначим $V, |V| = n$ – множество предприятий, которые требуется разместить для обслуживания клиентов. Пусть $P, |P| = n$ – множество позиций (точек), заданных на плоскости L, предназначенных для размещения предприятий из множества V, причем, очевидно, что в одну точку $k \in P$ возможно

разместить любое количество предприятий. Пусть $d(s,k)$ – расстояние между точками s и k, принадлежащих плоскости L; u_{ij} – неотрицательные удельные стоимости связей между размещаемым предприятием i и фиксированным клиентом j; w_{ik} – удельные стоимости связей между размещаемыми объектами i и k. Необходимо найти размещение предприятий из множества V, минимизирующее суммарную стоимость связи как между всеми предприятиями, так и между предприятиями и клиентами.

Ниже приведена формулировка задачи в терминах целочисленного линейного программирования. Пусть $x_t^i \in \{0,1\}$ булева переменная, где $x_t^i = 1$, если предприятие i размещено в позицию t, иначе $x_t^i = 0$. Пусть $z_{th}^{ij} \in \{0,1\}$ булева переменная, где $z_{th}^{ij} = 1$, если предприятия i и j размещены в позициях t и h соответственно, иначе $z_{th}^{ij} = 0$.

$$\sum_{i\in V}\sum_{t\in P} x_t^i \left(\sum_{j\in J} u_{ij} d(t, A_j)\right) + \sum_{i,j\in V}\sum_{t,h\in P} z_{th}^{ij} w_{ij} d(t,h) \to \min_{x,z} \tag{1.14}$$

при ограничениях

$$\sum_{t\in P} x_t^i = 1, \qquad i \in V; \tag{1.15}$$

$$\sum_{h\in P} z_{th}^{ij} = x_t^i, \qquad i,j \in V,\ t \in P; \tag{1.16}$$

$$\sum_{t\in P} z_{th}^{ij} = x_h^j, \qquad i,j \in V,\ h \in P; \tag{1.17}$$

$$x_t^i \in \{0,1\}, \qquad i \in V,\ t \in P; \tag{1.18}$$

$$z_{th}^{ij} \in \{0,1\}, \qquad i,j \in V,\ t,h \in P. \tag{1.19}$$

Легко заметить, что стоимость связи предприятия с клиентами однозначно определяется позицией размещения данного предприятия. Исходя из этого, в целевой функции (1.14) стоимость связи предприятия $i \in V$ с клиентами из

множества J определяется следующим образом: если предприятие i размещено в позицию t, то есть $x_t^i = 1$, то в целевую функцию включается стоимость $\sum_{j \in J} u_{ij} d(t, A_j)$. Ограничение (1.15) означает, что каждое предприятие $i \in V$ может быть размещено в одну и только одну позицию $t \in P$. Ограничения (1.16) и (1.16) говорят о том, что предприятия i и j, размещенные в соответствующие позиции t и h, могут быть связаны тогда и только тогда, когда соответствующие булевы переменные x_t^i и x_h^j равны 1.

§1.2. Класс задач о назначениях

Любая задача о назначениях может быть сформулирована следующим образом. Заданы два множества элементов, требуется найти отображение элементов одного множества в элементы другого с целью минимизации (максимизации) некоторого критерия. Различие в структурах заданных множеств, в типе отображения, в критериях оптимизации порождают различные вариации задач о назначениях. Все задачи о назначениях принято разделять по характеру целевой функции на два класса: линейные и нелинейные задачи о назначениях.

1.2.1. Линейные задачи о назначениях

Самым известным представителем данного класса задач является классическая *линейная задач о назначениях* [47]. Дано $V, |V| = n$ – множество заданий, которые необходимо распределить между работниками из множества $P, |P| = n$. Обозначим c_{ij} – стоимость назначения задания $i \in V$ работнику $j \in P$. Каждое задание должно быть назначено одному работнику, причем, каждый работник может выполнить только одно и только одно задание. Формулировка задачи в терминах линейного программирования имеет следующий вид:

$$\sum_{i \in V,\ j \in P} x_{ij} c_{ij} \to \min_x \tag{1.20}$$

при ограничениях

$$\sum_{j \in P} x_{ij} = 1, \qquad i \in V; \tag{1.21}$$

$$\sum_{i \in P} x_{ij} = 1, \qquad j \in V; \tag{1.22}$$

$$x_{ij} \geq 0, \qquad i \in V,\ j \in P. \tag{1.23}$$

Булева переменная $x_{ij} = 1$, если задание i назначено работнику j, иначе $x_{ij} = 0$. Ограничение (1.21) имеет смысл того, что каждое задание $i \in V$ может быть назначено одному и только одному работнику $j \in P$. Ограничение (1.22) означает, что каждый работник $j \in P$ может выполнить одно и только одно задание $i \in V$. Нетрудно заметить, что ограничения (1.21-1.22) вместе задают биективность отображения множества V в множество P.

Известно, что задача (1.20-1.23) разрешима за полиномиальное время с помощью известного венгерского алгоритма с трудоемкостью $O(n^3)$ [101]. Важно заметить, несмотря на то, что переменная x_{ij} непрерывна, она принимает только два значения: 0 и 1. Это происходит потому, что матрица ограничений рассматриваемой задачи является унимодулярной, то есть детерминант любой ее квадратной матрицы равен $\pm$ 1 или 0 [123].

Если в линейной задаче о назначениях (1.20-1.23) целевую функцию заменить на

$$\max_{i \in V,\ j \in P} \{x_{ij} c_{ij}\} \to \min,$$

то получим задачу, известную в зарубежной литературе как *Linear Bottleneck Assignment Problem* [47, 75]. Такая задача также полиномиально разрешима, и для ее решения был предложен точный алгоритм с трудоемкостью $O(n^2 \log n)$ [82].

Необходимо сказать, что существует множество других задач оптимизации, принадлежащих классу линейных задач о назначениях [47]. Большинство их имеют сходства с линейной задачей о назначениях и разрешимы за полино-

миальное время.

1.2.2. Нелинейные задачи о назначениях

Рассмотрим класс нелинейных задач о назначениях. Задачи, принадлежащие данному классу, являются существенно более трудным для решения, чем представители класса линейных задач о назначениях. Одним из широко известных представителей класса нелинейных задач о назначениях является *квадратичная задача о назначениях* (Quadratic Assignment Problem). Различают формулировки такой задачи в терминах матриц (постановка Купманса-Бекманна) [98] и в терминах теории графов [12]. Ниже приведена постановка с использованием терминов теории графов.

Задан неориентированный взвешенный граф G без петель и кратных ребер с множеством вершин $V(G), |V(G)| = n$ (размещаемые объекты) и множеством ребер $E(G)$ (связи между объектами). Каждому ребру $[i, j] \in E(G)$ присвоен вес $w(i, j) \geq 0$, равный удельной стоимости связей между объектами i и j. Дана неориентированная взвешенная сеть $M = (P, U)$, где $P, |P| = n$ – конечное множество позиций, предназначенных для размещения вершин графа G, U – коммуникации между позициями. Каждому ребру $[t, h] \in U$ присвоен вес $d(t, h) > 0$, равный расстоянию между позициями t и h. Размещением графа G на сети M будем называть биективное отображение $\pi : V(G) \to P$. Отображение π можно рассматривать как некоторую подстановку вершин графа G в вершины сети M. Пусть $b(i, t), b \geq 0$ – стоимость размещения вершины $i \in V(G)$ в позиции $t \in P$, и $c([i, j], t, h)$ – стоимость связи между вершинами $i, j \in V(G)$, размещенными в позициях $t, h \in P$.

Задача заключаются в следующем. Необходимо разместить вершины графа G в вершины сети M по одному в каждую таким образом, чтобы суммарная стоимость размещения вершин и суммарная стоимость связей между размещенными вершинами графа G была минимальной. Формулировка задачи в терми-

нах булевого программирования с *квадратичной целевой функцией* имеет вид:

$$\sum_{i\in V(G),\ t\in P} x_t^i \cdot b(i,t) + \sum_{[i,j]\in E(G),\ t,h\in P} x_t^i \cdot x_j^h \cdot \mathrm{c}([i,j],t,h) \to \min_x \tag{1.24}$$

при ограничениях

$$\sum_{t\in P} x_t^i = 1, \qquad i \in V(G); \tag{1.25}$$

$$\sum_{i\in V(G)} x_t^i = 1, \qquad t \in P; \tag{1.26}$$

$$x_t^i \in \{0,1\}, \qquad i \in V(G),\ t \in P. \tag{1.27}$$

Пусть x_t^i – булева переменная, где $x_t^i = 1$, если предприятие i размещено в позицию t, иначе $x_t^i = 0$. Ограничение (1.25) имеет смысл того, что каждый объект $i \in V(G)$ может быть размещен в одну и только одну позицию $t \in P$. Ограничение (1.26) означает, что в каждую позицию $t \in P$ может быть размещен один и только один объект $i \in V(G)$. Вместе ограничения (1.21-1.22) задают условие биективности отображения множества $V(G)$ в множество P.

Стоит отметить, что формулировка квадратичной задачи о назначениях может быть записана с использованием булевых переменных z_{th}^{ij}, где $z_{th}^{ij} = 1$, если предприятия i и j размещены в позициях t и h соответственно, иначе $z_{th}^{ij} = 0$.

$$\sum_{i\in V(G),\ t\in P} x_t^i \cdot b(i,t) + \sum_{[i,j]\in E(G),\ t,h\in P} z_{th}^{ij} \cdot \mathrm{c}([i,j],t,h) \to \min_{x,z} \tag{1.28}$$

$$\sum_{t\in P} x_t^i = 1, \qquad i \in V(G); \tag{1.29}$$

$$\sum_{i\in V(G)} x_t^i = 1, \qquad t \in P; \tag{1.30}$$

$$\sum_{h \in P} z_{th}^{ij} = x_t^i, \qquad [i,j] \in E(G),\ t \in P; \tag{1.31}$$

$$\sum_{t \in P} z_{th}^{ij} = x_h^j, \qquad [i,j] \in E(G),\ h \in P; \tag{1.32}$$

$$x_t^i \in \{0,1\}, \qquad i \in V(G),\ t \in P; \tag{1.33}$$

$$z_{th}^{ij} \in \{0,1\}, \qquad [i,j] \in E(G),\ t,h \in P. \tag{1.34}$$

Ограничение (1.29) имеет смысл того, что каждый объект $i \in V(G)$ может быть размещен в одну и только одну позицию $t \in P$. Ограничение (1.30) означает, что в каждую позицию $t \in P$ может быть размещен один и только один объект $i \in V(G)$. Вместе ограничения (1.29-1.30) задают условие биективности отображения множества $V(G)$ в множество P. Ограничения (1.31) и (1.32) говорят о том, что предприятия i и j, размещенные в соответствующие позиции t и h, могут быть связаны тогда и только тогда, когда соответствующие булевы переменные x_t^i и x_h^j равны 1.

Квадратичная задача о назначениях является, пожалуй, самым ярким представителем класса нелинейных задач о назначениях и была исследована множеством авторов. Важно отметить, что хорошо известная *задача о коммивояжере* [19,20,102] является частным случаем данной задачи. Для произвольных матриц и графов квадратичная задача является NP-трудной [126].

Вследствие NP-трудности задачи, одним из актуальных направлений ее исследования является поиск полиномиально разрешимых случаев [49]. Среди основных направлений исследования КЗН в терминах матриц можно выделить поиск сильно разрешимых случаев. Они представляют собой такие условия на матрицы, при которых решение задачи является заранее заданной подстановкой [6].

Для приближенного решения КЗН применяются известные эвристические методы: алгоритмы муравьиной колонии, имитации отжига, поиска с запретами, генетические и другие [26,56,59,62,71,104,127,133,134]. Основное внимание

в исследованиях КЗН в терминах теории графов уделяется разработке эффективных алгоритмов для специальных структур связей между объектами и сетей. Известны полиномиальные алгоритмы размещения изоморфных графов специальной структуры, эффективные алгоритмы размещения деревьев на цепи, алгоритм локальной оптимизации и другие [10, 11]. Предложен алгоритм динамического программирования решения задачи размещения произвольного взвешенного графа на невзвешенном дереве для минисуммного критерия. Для решения специальных случаев КЗН на сетях известны приближенные алгоритмы с оценками [29].

Следующим представителем класса нелинейных задач о назначениях является так называемая *мультииндексная задача о назначениях*, известная в зарубежной литературе как *Multi-Index Assignment Problem* [48]. В данной задаче требуется вычислить оптимальное назначение одновременно нескольких объектов. Например, трех-индексная задача состоит в оптимальном назначении заданий работникам, при одновременном нахождении оптимального назначения работников в позиции. Пусть I – множество заданий, J – множество работников и P – множество позиций, причем полагаем $|I| = |J| = |P| = n$. Пусть b_{ijp} стоимость назначения задания i работнику j, когда работник j размещен в позицию p. Обозначим y_{ijp} булеву переменную, принимающая значение 1, когда задание i назначено работнику j, а работник j назначен в позицию p, иначе $y_{ijp} = 0$. Формулировка задачи в терминах целочисленного линейного программирования имеет вид:

$$\sum_{i\in I}\sum_{j\in J}\sum_{p\in P} y_{ijp}b_{ijp} \to \min_{y} \tag{1.35}$$

при ограничениях

$$\sum_{i\in I}\sum_{j\in J} y_{ijp} = 1, \qquad p \in P; \tag{1.36}$$

$$\sum_{i \in I} \sum_{p \in P} y_{ijp} = 1, \qquad j \in J; \tag{1.37}$$

$$\sum_{j \in J} \sum_{p \in P} y_{ijp} = 1, \qquad i \in I; \tag{1.38}$$

$$y_{ijp} \in \{0, 1\}, \qquad i \in I,\ j \in J,\ p \in P. \tag{1.39}$$

Ограничение (1.36) означает, что в каждую позицию $p \in P$ может быть размещен один и только один работник $i \in J$. Ограничение (1.37) имеет смысл того, что работник $i \in J$ может взять одно и только одно задание $i \in I$ и может быть размещен в одну и только одну позицию $p \in P$. Ограничение (1.38) означает, что задание $i \in I$ может быть назначено одному и только одному работнику $i \in J$.

Известно, что трех-индексная задача о назначениях, так же как и ее обобщение мультииндексная задача о назначениях, являются NP-трудными [48]. Для решения данных задач предложены алгоритмы ветвей и границ, эвристики и метаэвристики [34, 50, 50, 107, 108]. Для некоторых частных случаев мультииндексной задачи о назначениях предложены полиномиальные приближенные схемы [35, 63].

Следующий представитель класса нелинейных задач о назначения – задача, известная в зарубежной литературе как *dynamic facility layout problem* [60]. Дан неориентированный взвешенный граф G с множеством вершин $V(G), |V(G)| = n$ и множеством ребер $E(G)$. Каждому ребру $[i, j] \in E(G)$ присвоен вес $w(i, j) \geq 0$. Дана неориентированная взвешенная сеть $M = (P, U)$, где $P, |P| = m,\ m \leq n$ – конечное множество позиций, предназначенных для размещения вершин графа G. Каждому ребру $[t, h] \in U$ присвоен вес $d(t, h) > 0$. Для каждой вершины t сети M задана величина $a_t,\ a_t \geq 1$, равная числу вершин графа G, которые требуется назначить вершине t. Размещением графа G на сети M будем называть отображение $\pi : V(G) \to P$, причем в данном случае в одну позицию возможно

разместить несколько вершин. Пусть $b(i,t), b \geq 0$ – стоимость размещения вершины $i \in V(G)$ в позиции $t \in P$, и $c([i,j],t,h)$ – стоимость связи между вершинами $i,j \in V(G)$, размещенными в позициях $t,h \in P$. Необходимо разместить вершины графа G в вершины сети M, чтобы суммарная стоимость размещения вершин и суммарная стоимость связей между размещенными вершинами графа G была минимальной. Формулировка задачи имеет вид:

$$\sum_{i \in V(G),\ t \in P} x_t^i \cdot b(i,t) + \sum_{[i,j] \in E(G),\ t,h \in P} z_{th}^{ij} \cdot \mathrm{c}([i,j],t,h) \to \min_{x,z} \tag{1.40}$$

при ограничениях

$$\sum_{t \in P} x_t^i = 1, \qquad i \in V(G); \tag{1.41}$$

$$\sum_{i \in V(G)} x_t^i = a_t, \qquad t \in P; \tag{1.42}$$

$$\sum_{h \in P} z_{th}^{ij} = x_t^i, \qquad [i,j] \in E(G),\ t \in P; \tag{1.43}$$

$$\sum_{t \in P} z_{th}^{ij} = x_h^j, \qquad [i,j] \in E(G),\ h \in P; \tag{1.44}$$

$$x_t^i \in \{0,1\}, \qquad i \in V(G),\ t \in P; \tag{1.45}$$

$$z_{th}^{ij} \in \{0,1\}, \qquad [i,j] \in E(G),\ t,h \in P. \tag{1.46}$$

Булева переменная $x_t^i = 1$, если предприятие i размещено в позицию t, иначе $x_t^i = 0$. Булева переменная $z_{th}^{ij} = 1$, если предприятия i и j размещены в позициях t и h соответственно, иначе $z_{th}^{ij} = 0$. Ограничение (1.41) имеет смысл того, что каждый объект $i \in V(G)$ может быть размещен в одну и только одну позицию $t \in P$. Ограничение (1.42) означает, что в каждую позицию $t \in P$ должно быть размещено ровно a_t объектов из множества $V(G)$. Очевидно, что в данной задаче условие биективности отображения множества $V(G)$ в множество

P не выполняется. Ограничения (1.43) и (1.44) говорят о том, что предприятия i и j, размещенные в соответствующие позиции t и h, могут быть связаны тогда и только тогда, когда соответствующие булевы переменные x_t^i и x_h^j равны 1.

Очевидно, что задача (1.40-1.46) является NP-трудной. Для ее решения предложены алгоритмы ветвей и границ, эвристики и метаэвристики, а также эффективные алгоритмы вычисления нижних границ задачи [60, 129, 135].

Легко заметить, что данная задача повторяет квадратичную задачу о назначениях, за исключением следующих условий в модели целочисленного линейного программирования. Первое отличие – соотношение числа вершин графа G к числу вершин сети M: в квадратичной задаче о назначениях выполняется $|V(G)| = |P|$, а в рассматриваемой задаче справедливо $|V(G)| \geq |P|$. Второе отличие – ограничение (1.26) в квадратичной задаче о назначениях заменяется в данной задаче на ограничение (1.42).

§1.3. Формулировка исследуемой задачи

Постановка дискретной задачи Вебера (ДЗВ) в терминах теории графов следующая. Пусть G – простой взвешенный связный граф, $V(G)$ – множество вершин графа G, соответствующее размещаемым объектам, $E(G)$ – множество ребер графа G, определяющее связи между размещаемыми объектами. Пусть P – конечное множество позиций, предназначенных для размещения вершин графа G. В дальнейшем будем полагать $|V| = |P| = n$, что не ограничивает общности изложения. Размещением вершин графа G будем называть отображение $\pi : V(G) \to P$, полагая, что вершина $i \in V(G)$ размещается в позицию $\pi(i) \in P$, причем в одну позицию возможно разместить несколько вершин. Множество всех отображений множества вершин $V(G)$ в множество позиций P обозначим через $\Pi = \{\pi : V(G) \to P\}$.

Заданными являются: вес ребра $w(i, j) \geq 0$ для $[i, j] \in E(G)$; расстояние $d(t, u) > 0$ между позициями $t, u \in P$, определенное в некоторой метрике; сто-

имость размещения $b(i,t) \geq 0$ вершины $i \in V(G)$ в позиции $t \in P$; стоимость связи $c([i,j],t,u)$ между вершинами $i,j \in V(G)$, размещенными в позициях $t,u \in P$, причем полагаем $c([i,j],t,u) = w(i,j) \cdot d(t,u)$. Необходимо разместить вершины графа G в позициях P таким образом, чтобы суммарная стоимость размещения вершин и ребер графа G была минимальной. Формулировка задачи в терминах отображений следующая:

$$F(G,\pi) = \sum_{[i,j]\in E(G)} c([i,j],\pi(i),\pi(j)) + \sum_{i\in V(G)} b(i,\pi(i)) \to \min_{\pi\in\Pi}. \tag{1.47}$$

Имеет место следующая формулировка ДЗВ в терминах целочисленного линейного программирования. Пусть x_t^i – булева переменная, где $x_t^i = 1$, если предприятие i размещено в позицию t, иначе $x_t^i = 0$. Пусть z_{th}^{ij} – булева переменная, где $z_{th}^{ij} = 1$, если предприятия i и j размещены в позициях t и h соответственно.

$$\sum_{i\in V(G),\ t\in P} x_t^i b(i,t) + \sum_{[i,j]\in E(G),\ t,h\in P} z_{th}^{ij} \mathrm{c}([i,j],t,h) \to \min_{x,z} \tag{1.48}$$

при ограничениях

$$\sum_{t\in P} x_t^i = 1, \qquad i \in V(G); \tag{1.49}$$

$$\sum_{h\in P} z_{th}^{ij} = x_t^i, \qquad [i,j] \in E(G),\ t \in P; \tag{1.50}$$

$$\sum_{t\in P} z_{th}^{ij} = x_h^j, \qquad [i,j] \in E(G),\ h \in P; \tag{1.51}$$

$$x_t^i \in \{0,1\}, \qquad i \in V(G),\ t \in P; \tag{1.52}$$

$$z_{th}^{ij} \in \{0,1\}, \qquad [i,j] \in E(G),\ t,h \in P. \tag{1.53}$$

Ограничение (1.49) имеет смысл того, что каждый объект $i \in V(G)$ может быть размещен в одну и только одну позицию $t \in P$. Очевидно, что в

рассматриваемой задаче условие биективности отображения множества $V(G)$ в множество P не выполняется, поскольку ограничение на количество размещаемых в позицию объектов отсутствует. Ограничения (1.50) и (1.51) говорят о том, что предприятия i и j, размещенные в соответствующие позиции t и h, могут быть связаны тогда и только тогда, когда соответствующие булевы переменные x_t^i и x_h^j равны 1.

Дискретная задача Вебера, состоящая в размещении графа G на множестве P с критерием F в дальнейшем будет обозначаться тройкой $\langle G, P, F\rangle$. Известно, что задача $\langle G, P, F\rangle$ является NP-трудной [14,126]. В зарубежной литературе она носит названия *quadratic semi-assignment problem* и *task assignment problem*. Для случаев задачи, когда размещаемый граф является деревом [44, 116] и последовательно-параллельным графом [110] известны точные полиномиальные алгоритмы. Также случай, когда $|P| = 2$, разрешим за полиномиальное время [130]. Предложены эффективные алгоритмы для вычисления нижней границы оптимума задачи [112]. Известны эффективные эвристики и метаэвристики [97]. Для ДЗВ в общем виде, впрочем, так же как и для квадратичной задачи о назначениях в общей постановке, доказана ее неаппроксимируемость [126]. Несмотря на это, для частного случая, когда

$$(\forall [i,j] \in E(G),\ \forall t,u \in P) \quad c([i,j],t,u) = \begin{cases} 0, \text{ если } t = u; \\ \alpha, \text{ если } t \neq u, \end{cases}$$

где $\alpha = const$, найдена полиномиальная приближенная схема [70].

Легко показать связь между ДЗВ и задачей (1.14-1.19), являющейся дискретным вариантом задачи *MFW*. Действительно, поскольку в формуле (1.14) значение выражения $\sum_{j\in J} u_{ij} d(t, A_j)$ стоимости связи предприятия i с клиентами из множества J полностью определяется позицией $t \in P$ размещения

предприятия i, то полагая в формуле (1.14)

$$\sum_{j \in J} u_{ij} d(t, A_j) = b(i, t),$$

целевая функция (1.14) будет повторять целевую функцию (1.48) дискретной задачи Вебера, притом, что ограничения в обоих задачах сходны. Важно заметить, что ДЗВ имеет существенное преимущество перед задачей (1.14-1.19), поскольку позволяет учитывать в значении функции стоимости $b(\cdot)$ не только стоимость связи предприятия с клиентами, но также и стоимость размещения (постройки) предприятия в позиции.

Дискретная задача Вебера так же может быть сформулирована как задача нелинейного программирования. Пусть x_t^i – булева переменная, где $x_t^i = 1$, если предприятие i размещено в позицию t, иначе $x_t^i = 0$.

$$\sum_{i \in V(G),\ t \in P} x_t^i \cdot b(i,t) + \sum_{[i,j] \in E(G),\ t,h \in P} x_t^i \cdot x_j^h \cdot \mathrm{c}([i,j],t,h) \to \min_x \tag{1.54}$$

при ограничениях

$$\sum_{t \in P} x_t^i = 1, \qquad i \in V(G); \tag{1.55}$$

$$x_t^i \in \{0,1\}, \qquad i \in V(G),\ t \in P. \tag{1.56}$$

В такой формулировке нетрудно заметить сходство ДЗВ с квадратичной задачей о назначениях. Единственное отличие ДЗВ от квадратичной задачи о назначениях заключается в том, что ограничение (1.26) в модели ДЗВ отсутствует, что позволяет разместить любое число объектов в позиции. Такое различие между ДЗВ и квадратичной задачей о назначениях наглядно проиллюстрировано на рисунке 1.3.

Как показано на рисунке, в ДЗВ возможен случай, когда в одну позицию

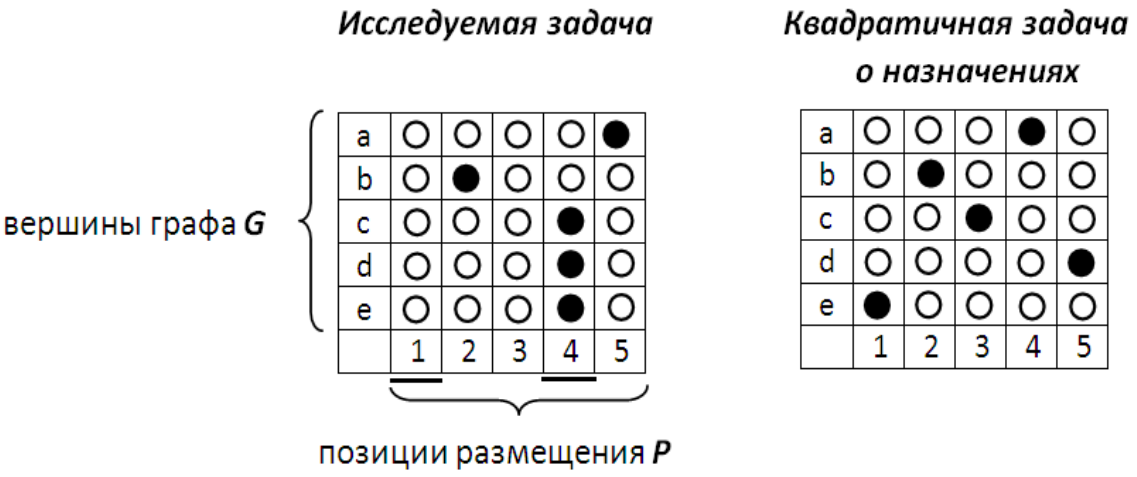

Рис. 1.3. Различие между ДЗВ и квадратичной задачей о назначениях

размещено несколько объектов, а также случай, когда в позицию не размещено объектов вовсе, в то время как в квадратичной задаче о назначениях в каждой позиции размещается ровно по одному объекту.

Нетрудно заметить, что если в модель целочисленного программирования ДЗВ добавить ограничение

$$\sum_{i \in V(G)} x_t^i = 1, \qquad t \in P,$$

то получившаяся задача будет полностью повторять квадратичную задачу о назначениях.

Области практических приложений дискретной задачи Вебера достаточно разнообразны. ДЗВ является базовой математической моделью во многих прикладных задачах, возникающих, например, в при решении задач управления вычислительными ресурсами в распределенных гетерогенных вычислительных средах, а именно для решения задачи оптимального размещении вычислительных заданий в узлах распределенной вычислительной среды; автоматизации проектирования схем расположения взаимосвязанных объектов в заданной области; размещение промышленного

предприятия, состоящего из нескольких объектов (перерабатывающие заводы, склады, торговые точки) и т.д. Важная черта ДЗВ – допущение размещения несколько объектов в одну позицию, что порой является критически важным условием при решении реальных прикладных задач, когда технологически разрешается размещение нескольких предприятий в одном населенном пункте (городе). Заметим, что хорошо известная в литературе квадратичная задача о назначениях такой возможности не допускает.

§1.4. Выводы по первой главе

В первой главе приведен обзор известных в литературе непрерывных постановок задачи Вебера. Приведены их формулировки в терминах частично целочисленного линейного программирования. Приведен краткий обзор известных в литературе результатов, полученных для данных задач. Рассмотрены классы линейных и нелинейных задач о назначениях, и приведен перечень задач, являющихся наиболее яркими представителями таких классов. Приведена формулировка исследуемой в работе задачи в терминах отображений и целочисленного линейного программирования. Показана связь исследуемой задачи с дискретным вариантом непрерыной задачи MFW и с квадратичной задачей о назначениях. Приведен обзор известных в литературе результатов, полученных другими авторами для исследуемой задачи. Дан обзор направлений практического применения исследуемой задачи.

Глава 2.

Точные алгоритмы решения задачи для графов специального вида

В настоящей главе предлагаются точные полиномиальные алгоритмы решения ДЗВ для специальных структур связей между размещаемыми объектами. Все алгоритмы, предложенные в первой главе, основаны на идее динамического программирования. В разделе 2.1 предлагается алгоритм решения ДЗВ для k-дерева с трудоемкостью $O(n^{k+3})$ и оценкой требуемой оперативной памяти $O(n^{k+3})$. Предложенный алгоритм для k-дерева обобщен на класс графов с ограниченным параметром древовидной ширины. В разделе 2.2 приводится алгоритм решения ДЗВ для k-цепи с трудоемкостью $O(n^{k+2})$ и оценкой требуемой оперативной памяти $O(n^k)$. Предложенный алгоритм для k-цепи обобщен на класс графов с ограниченным параметром ленточной ширины. В разделе 2.3 представлен алгоритм решения ДЗВ для простого цикла с трудоемкостью $O(n^4)$ и оценкой требуемой оперативной памяти $O(n^2)$.

§2.1. Точный алгоритм решения задачи для k-дерева

Рассматривается задача $\langle T, P, F\rangle$, где T – есть k-дерево. Доказывается, что задача $\langle T, P, F\rangle$ полиномиально разрешима при фиксированном параметре k. Предлагается точный алгоритм A_T решения задачи $\langle T, P, F\rangle$, основанный на динамическом программировании по дереву декомпозиции. Предложенный алгоритм опубликован в работе [17].

2.1.1. Определение k-дерева. Дерево декомпозиции k-дерева

Приведем известное определение k-дерева [124].

Определение 1.1. *Связный неориентированный граф T называется k-деревом, если его построение возможно осуществить рекурсивно по правилам: полный граф из $k+1$ вершин есть k-дерево; k-дерево с $i+1$ вершинами получается из k-дерева с i вершинами добавлением в него новой вершины j и k ребер таким образом, чтобы новая вершина j стала смежной со всеми вершинами некоторой k-клики (клики размера k).*

Класс k-деревьев был введен в 70-х годах прошлого столетия в работах [124, 125]. Интерес к изучению графов такого вида вызван тем, что некоторые NP-трудные задачи теории графов полиномиально разрешимы, если модельный граф является k-деревом. В данных условиях задача может быть решена методом динамического программирования на основе *дерева декомпозиции* модельного графа [122]. На сегодняшний день известны точные полиномиальные алгоритмы решения *классических задач теории графов* на k-деревьях [30], например, задачи о максимальной клике, задачи о минимальном доминирующем множестве, задачи Штейнера и др. Также известны полиномиальные алгоритмы для некоторых *задач размещения* на k-деревьях [31]: простейшей задачи размещения, задачи о p-медиане и др.

Ниже приведены некоторые свойства k-дерева, используемые в настоящей работе. Если граф T есть k-дерево с n вершинами, то

1. *Все максимальные клики в T имеют размер k;*
2. *Мощность минимального сепаратора в T равна k;*
3. *Количество ребер k-дерева T равно $nk - k(k+1)/2$;*
4. *Количество $(k+1)$-клик в T равно $n-k$;*
5. *Количество k-клик в T равно $k(n-k)+1$;*

6. *Число вершинно непересекающихся путей в T, связывающих любые две несмежные вершины, равно k.*

Рекурсивное определение k-дерева предполагает, что графы данного класса обладают свойством *совершенного порядка элиминации*, известного в зарубежной литературе, как *perfect elimination order property* [78]. То есть для k-дерева T с n вершинами существует такая нумерация вершин $v_1, v_2, ..., v_j, ..., v_n$, что в подграфе k-дерева T, индуцированном множеством вершин $V(T) \setminus \{v_j : j = 1, 2, ..., i-1\}$, где $i \geq 1$, вершины, смежные с вершиной v_i образуют клику размера k. На рисунке 2.1 представлены k-деревья для различных значений k. Подробный анализ свойств k-дерева можно найти в работах [114, 124].

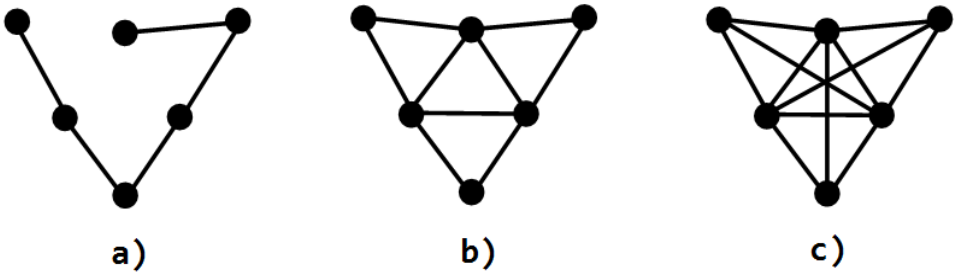

Рис. 2.1. k-деревья: a) 1-дерево; b) 2-дерево; c) 3-дерево

Дадим следующее определение дерева декомпозиции k-дерева, удобное для описания предлагаемого алгоритма решения рассматриваемого частного случая ДЗВ. Деревом декомпозиции k-дерева T в дальнейшем будем называть дерево $\mathcal{T} = (\mathcal{M}, \mathcal{W})$, с множеством узлов $\mathcal{M} = K \cup S$, где K есть множество $(k+1)$-клик графа T, множество $S = \{X \cap Z : \ X, Z \in K, |X \cap Z| = k\}$, и множеством ребер $\mathcal{W} = \{[X, Y] : X \in K, Y \in S, |X \cap Y| = k\}$. Вершины дерева $\mathcal{T}$ будем называть узлами во избежание путаницы с вершинами исходного k-дерева T. Узел $X \in K$ в дальнейшем будем называть *«узлом-кликой»*. Поскольку каждое множество $Y \in S$ является сепаратором графа T [1, 2] (то есть при удалении вершин, принадлежащих $Y \in S$, а также всех смежных с ними ребер из k-дерева T, увеличивается число компонент связности графа T),

то соответствующий узел Y будем называть *«узлом-сепаратором»*. Стоит заметить, что ребро $[X, Y] \in \mathcal{W}$ существует только между узлом-кликой $X \in K$ и узлом-сепаратором $Y \in S$, и только тогда, когда $Y \subset X$ [90].

Пример 2-дерева и его дерева декомпозиции представлены на рис. 2.2. Поскольку любое k-дерево есть хордальный граф, то для построение дерева де-

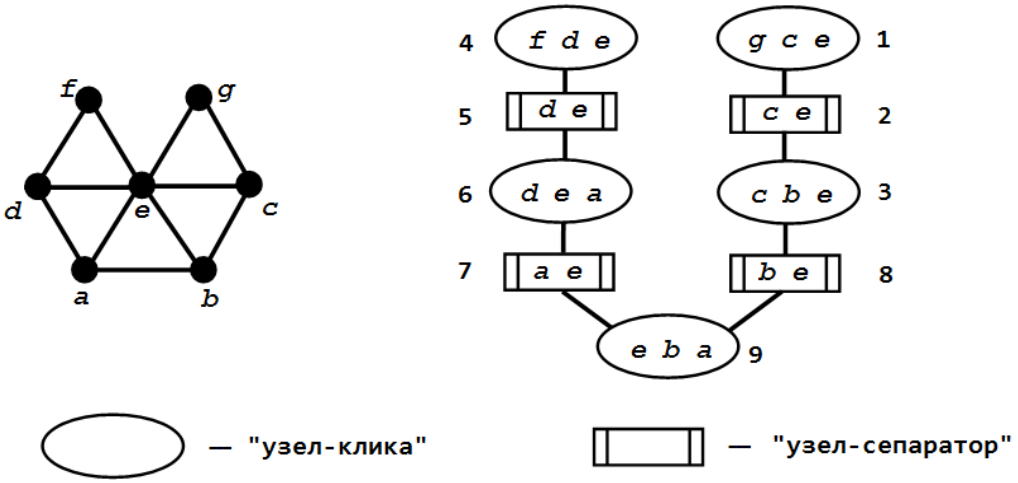

Рис. 2.2. Дерево декомпозиции 2-дерева

композиции k-дерева T могут быть использованы известные полиномиальные алгоритмы, например, алгоритм [90], имеющий оценку времени работы $O(n^3)$. Обозначим $N(i)$ – множество вершин графа, смежных с вершиной i. Для полноты изложения, ниже приведем описание алгоритма [90].

Положить $\tilde{T} = T$.

Выполнять шаги 1-4, пока $|\tilde{T}| > k + 1$.

Шаг 1. Найти вершину $i \in V(\tilde{T})$ такую, что $N(i) = k$.

Шаг 2. Включить клику $i \cup N(i)$ размера $k + 1$ в множество K.

Шаг 3. Включить клику $N(i)$ размера k в множество S.

Шаг 4. Построить новый граф $\tilde{T}$ путем удаления вершины i и всех смежных с ней ребер.

Если $|\tilde{T}| = k + 1$, то выполнить шаги 5-6, иначе повторить шаги 1-4.

Шаг 5. Множество вершин $V(\tilde{T})$ оставшегося полного графа включить в K.

Шаг 6. Определить $\mathcal{M} = K \cup S$ множество узлов и множество ребер $\mathcal{W} = \{[X, Y] : X \in K,\, Y \in S,\, |X \cap Y| = k\}$ дерева декомпозиции $\mathcal{T}$.

2.1.2. Точный алгоритм размещения k-дерева

Для ясности описания логики работы алгоритма A_T введем следующие обозначения. Пусть N – мощность множества $\mathcal{M}$. Выбор узла $X_N \in \mathcal{M}$ в качестве корня дерева $\mathcal{T}$ индуцирует на множестве узлов $\mathcal{M}$ отношение частичного порядка

$$L = \{(X, Y) : Y \text{ принадлежит цепи в } \mathcal{T} \text{ между } X \text{ и } X_N\}, \qquad (2.1)$$

где $X, Y \in \mathcal{M}$. В дальнейшем будем считать, что $\mathcal{M} = \{X_i\}_{i=1}^{N}$ и выполняется условие $(X_l, X_m) \in L \Rightarrow l < m$ (см. рис. 2.2). Для каждого узла $X_i \in \mathcal{M}$ определяется D_i – множество узлов, являющихся *прямыми потомками* узла X_i, то есть для любого $X_j \in D_i$ ребро $[X_i, X_j] \in \mathcal{W}$ и $i > j$. Также для каждого узла $X_i \in \mathcal{M}$ определяется $\bar{D}_i$ – множество узлов, являющихся *потомками* узла X_i, то есть для каждого $X_j \in \bar{D}_i$ найдется такая цепь $l = \{X_i, ..., X_j\}$ соединяющая узлы X_i и X_j, что для любого $X_m \in l, X_m \neq X_i$ справедливо неравенство $i > m$.

Пусть $\Pi(X_i) = \{\pi_i : X_i \to P\}$ – множество всех отображений вершин клики $X_i \in \mathcal{M}$ в множестве позиций P, где соответственно $\pi_i = \{\pi_i(j) : j \in X_i\}$ – некоторый способ размещения вершин клики X_i в множестве позиций P. Отметим, что для любых узлов-клик $X_i \in K$ имеет место равенство $|\Pi(X_i)| = n^{k+1}$, а для любых узлов-сепараторов $Y_i \in S$ справедливо равенство $|\Pi(Y_i)| = n^k$. Отношение между размещениями π_i и π_j, когда справедливо равенство $|\pi_i \cap \pi_j| = k$, будем обозначать $\pi_i \bowtie \pi_j$. Пусть $B(\pi_i)$ – функция стоимости способа

размещения π_i вершин клики X_i в множестве позиций P,

$$B(\pi_i) = \sum_{j \in X_i} b(j, \pi_i(j)).$$

Пусть $C(\pi_i)$ – функция стоимости размещения ребер подграфа графа T, индуцированного множеством вершин X_i, когда размещение вершин клики X_i в позициях P равно π_i,

$$C(\pi_i) = \sum_{j,l \in X_i} c([j,l], \pi_i(j), \pi_i(l)).$$

Обозначим $T(\pi_i)$ – подграф графа T, индуцированный вершинами, принадлежащими множеству $X_i \cup \bar{D}_i$, в котором размещение вершин его клики X_i в P равно π_i. Множество состояний процесса ДП на шаге i будем полагать равным $\Pi(X_i)$. Значение функции Беллмана $f_i(\pi_i)$, вычисленное для некоторого состояния $\pi_i \in \Pi(X_i)$, есть стоимость оптимального размещения подграфа $T(\pi_i)$ в множестве позиций P.

АЛГОРИТМ A_T

Вход: k-дерево T; множество позиций P; функции $b(\cdot)$ и $c(\cdot)$.

Выход: оптимальное отображение π^* вершин графа T в P.

Прямой ход.

Шаг 0. Построить дерево декомпозиции $\mathcal{T} = (\mathcal{M}, \mathcal{W})$ k-дерева T. Выбрать любой узел-клику $X_N \in \mathcal{M}$ в роли корня дерева $\mathcal{T}$. Задать отношение порядка (2.1) на множестве узлов $\mathcal{M}$.

Шаг i, $i = 1, ..., N$.

Если X_i есть узел-клика, то для каждого $\pi_i \in \Pi(X_i)$ выполнить:

$$f_i(\pi_i) = B(\pi_i) + C(\pi_i) + \sum_{X_j \in D_i : \pi_j \bowtie \pi_i} f_j(\pi_j) - \sum_{X_j \in D_i : \pi_j \bowtie \pi_i} \left(B(\pi_j) + C(\pi_j)\right), \quad (2.2)$$

где четвертое слагаемое правой части формулы используется для исключения «повторного счета» при сложении стоимостей размещения вершин и ребер подграфа, индуцированного кликой X_i, и подграфов $T_j,\ X_j \in D_i$;

$$M(\pi_i) = \{\pi_j : X_j \in D_i,\ \pi_j \bowtie \pi_i\}.$$

Если X_i есть узел-сепаратор, то для каждого $\pi_i \in \Pi(X_i)$ выполнить:

$$f_i(\pi_i) = \sum_{X_j \in D_i} \min_{\pi_j \in \Pi(X_j):\pi_j \bowtie \pi_i} \{f_j(\pi_j)\} - (B(\pi_i) + C(\pi_i)) \cdot (|D_i| - 1); \tag{2.3}$$

$$M(\pi_i) = \left\{ \pi_j^* : X_j \in D_i,\ \pi_j^* = \arg \min_{\pi_j \in \Pi(X_j):\pi_j \bowtie \pi_i} f_j(\pi_j) \right\}.$$

Шаг $N+1$. Вычислить стоимость f^* оптимального размещения вершин графа T в P по формуле

$$f^* = \min_{\pi_N \in \Pi(X_N)} \{f_N(\pi_N)\}. \tag{2.4}$$

Обратный ход.

Положить $\pi^* = \varnothing$, $i = N$ и $\pi_N^* = \arg\min_{\pi_N \in \Pi(X_N)} \{f_N(\pi_N)\}$.

Шаг 1. Включить π_i^* в множество π^*. Для всех $X_j \in D_i$ определить $\pi_j^* = \pi_j : \pi_j \in M(\pi_i^*)$. Положить $i = i - 1$. Если $i > 1$, то повторить шаг 1, иначе **стоп**.

Конец алгоритма.

Теорема 2.1. *Алгоритм A_T за время $O(n^{k+3})$ находит точное решение n-вершинной задачи $\langle T, P, F\rangle$. Оценка требуемой алгоритмом оперативной памяти равна $O(n^{k+3})$.*

Доказательство. Докажем, что размещение π^*, найденное алгоритмом A_T, оптимально. Обозначим $\langle T(\pi_i), P, F\rangle$ подзадачу, заключающуюся в оптимальном размещении вершин подграфа $T(\pi_i)$ в множестве позиций P.

Любая подзадача $\langle T(\pi_i), P, F\rangle,\ \pi_i \in \Pi(X_i)$, когда X_i есть узел-клика, решается оптимально, поскольку, в силу того, что в одну позицию возможно

разместить несколько вершин графа, оптимальная стоимость $f_i(\pi_i)$ размещения графа $T(\pi_i)$ может быть вычислена путем сложения стоимости размещения клики π_i и соответствующих стоимостей оптимального размещения подграфов $T(\pi_j),\ X_j \in D_i$, вычисленных на предыдущих шагах. В свою очередь, каждая подзадача $\langle T(\pi_i), P, F\rangle$, когда X_i есть узел-сепаратор, в силу того, что в одну позицию возможно разместить несколько вершин графа, также решается оптимально, поскольку стоимость $f_i(\pi_i)$ размещения графа $T(\pi_i)$ вычисляется по формуле (2.3), а величина $f_j(\pi_j)$, для любых $\pi_j \in \Pi(X_j),\ X_j \in D_i$, есть минимум подзадачи $\langle T(\pi_j), P, F\rangle$.

Таким образом, рекуррентно решая подзадачи $\langle T(\pi_i), P, F\rangle$ на шагах $i = 1, ..., N$, алгоритм на шаге N, где X_N – корень дерева $\mathcal{T}$, для каждого $\pi_N \in \Pi(X_N)$ находит оптимальное решение соответствующей подзадачи $\langle T(\pi_N), P, F\rangle$. Исходя из этого, алгоритм на конечном шаге $N + 1$, согласно формуле (2.4), находит оптимальное решение π^* исходной задачи $\langle T, P, F\rangle$.

Докажем оценки требуемой оперативной памяти и времени работы алгоритма. На шаге i, где X_i есть узел-клика, необходимо $O(n^{k+2})$ операций, поскольку для каждого состояния $\pi_i \in \Pi(X_i)$ вычисление значения функций $f_i(\pi_i)$ и $M(\pi_i)$ требует не более $O(n)$ операций, а число состояний равно n^{k+1}. На шаге i, где X_i есть узел-сепаратор, также требуется $O(n^{k+2})$ операций, так как для каждого состояния $\pi_i \in \Pi(X_i)$ трудоемкость вычисления значения функций $f_i(\pi_i)$ и $M(\pi_i)$ не превосходит $O(n^2)$ операций, а число состояний равно n^k. Поскольку число вершин в дереве декомпозиции $\mathcal{T}$ не превосходит $O(n)$, то *оценка времени работы* алгоритма A_T равна $O(n^{k+3})$.

Положим, что для хранения одного множества $M(\cdot)$ требуется $O(n)$ памяти, поскольку, для хранения одного элемента множества требуется $O(1)$ памяти, так как нам достаточно хранить лишь номер соответствующей ячейки в массиве $\Pi_j,\ j \in D_i$, а количество узлов-потомков вершины i не превосходит n. На шаге i, где X_i есть узел-клика, необходимо $O(n^{k+2})$ памяти, поскольку

$O(n^{k+1})$ памяти требуется для хранения значений массива $f_i(\cdot)$ и $O(n^{k+2})$ памяти требуется для хранения значений массива $M(\cdot)$. На шаге i, где X_i есть узел-сепаратор, необходимо $O(n^{k+1})$ памяти, поскольку $O(n^k)$ памяти требуется для хранения значений массива $f_i(\cdot)$ и $O(n^{k+1})$ памяти требуется для хранения значений массива $M(\cdot)$. Так как в общем случае, на каждом шаге i требуется хранить не более чем $O(n)$ массивов $f_i(\cdot)$ и $M(\cdot)$ вычисленных на предыдущих шагах, то *оценкой требуемой оперативной памяти* алгоритма A_T не превосходит $O(n^{k+3})$. **Теорема 2.1 доказана.**

Стоит заметить, что при любом фиксированном значении входного параметра k трудоемкость алгоритма A_T полиномиальна и ограничена полиномом степени $k+3$. Отсюда следует, что практическое применение предложенного алгоритма, впрочем, как и любых алгоритмов, использующих идею динамического программирования по дереву декомпозиции, ограничено большими затратами вычислительных операций и высокой потребностью в памяти, в тех случаях, когда величина параметра k сравнительно велика.

Вследствие экспоненциального возрастания трудоемкости предложенного алгоритма, очевидно, что применение его для решения дискретной задачи Вебера перспективно при сравнительно малых значениях величины параметра k, причем, чем меньше величина k и чем больше количество вершин размещаемого графа, тем предлагаемый алгоритм более эффективен по сравнению с точными алгоритмами, являющимися вариациями полного перебора с отсевом заведомо бесперспективных подмножеств допустимых решений, например алгоритма ветвей и границ, реализованного в пакете IBM ILOG CPLEX, так как оценка времени работы алгоритма A_T ограничивается полиномом $O(n^{k+3})$, в то время как вычислительная сложность таких точных алгоритмов в общем случае экспоненциальна. Очевидно, что чем ближе величина параметра k к числу вершин размещаемого графа, тем предлагаемый алгоритм менее эффективен по сравнению с такими алгоритмами, являющимися вариациями полного пе-

ребора. Например, при решении дискретной задачи Вебера для k-дерева T с количеством вершин $|V(T)| = k + 1$ время работы предложенного алгоритма A_T сравнимо с временем работы алгоритма, использующего идею тривиального последовательного перебора всех допустимых решений.

2.1.3. Обобщение алгоритма для k-дерева на класс графов с ограниченной древовидной шириной

Покажем, что из факта существования точного алгоритма решения ДЗВ для k-дерева с трудоемкостью $O(n^{k+3})$ следует, что любая ДЗВ для графа с параметром древовидной ширины равным k также может быть разрешена за время, ограниченное полиномом степени $k + 3$.

Приведем необходимые определения.

Определение 1.2 [122]. *Дерево декомпозиции произвольного графа G представляет собой пару $(\mathcal{X}, \mathcal{T})$, где $\mathcal{X} = \{X_i : i \in V(\mathcal{T})\}$ – семейство подмножеств множества $V(G)$ и $\mathcal{T}$ – дерево, каждой вершине i которого сопоставлено множество $X_i \in \mathcal{X}$ так, что выполняются следующие условия:*

1. $\bigcup_{i \in V(\mathcal{T})} X_i = V(G)$.
2. *Для каждого $[t, q] \in E(G)$ найдется $i \in V(\mathcal{T})$, что $t \in X_i$ и $q \in X_i$.*
3. *Для любых вершин $y, j, z \in V(\mathcal{T})$ справедливо: если j принадлежит цепи, связывающей вершины y и z в дереве $\mathcal{T}$, то $X_y \cap X_z \subseteq X_j$.*

Определение 1.3 [122]. *Параметр $tw(G)$ древовидной ширины (treewidth) графа G – есть число, равное наименьшей древовидной ширине всех его возможных деревьев декомпозиции, где древовидная ширина некоторого дерева декомпозиции $(\mathcal{X}, \mathcal{T})$ равна* $\max_{i \in V(\mathcal{T})} |X_i| - 1$.

Нетрудно заметить, что древовидная ширина отражает, насколько «близок» граф G к дереву и, очевидно, любое дерево имеет древовидную ширину, равную

1. Параметр древовидной ширины графа является для некоторых NP-трудных графовых задач мерилом сложности их решения методом динамического программирования [2, 122].

Классом графов с ограниченной древовидной шириной будем называть множество всех графов, для которых параметр их древовидной ширины меньше либо равен k, где k – положительная константа, не зависящая от n. Так, поскольку для всех деревьев параметр древовидной ширины 1 и не зависит от числа вершин n, то класс деревьев является классом графов с ограниченной древовидной шириной. Аналогично, поскольку для всех последовательно-параллельных графов параметр древовидной ширины ≤ 2 и не зависит от числа вершин n, то класс последовательно-параллельных графов является классом графов с ограниченной древовидной шириной.

Однако ограниченная древовидная ширина свойственна не всем графам. В частности, $tw(K_n) = n - 1$, где K_n – клика размера n. Кроме того, существуют совсем простые по структуре графы, для которых значение $tw(G)$ растет с увеличением числа вершин, то есть при $n \to \infty$ подобные графы все больше отдаляются от дерева. Типичным примером являются «сетки» – графы, построенные из правильных многогранников так, что любые два соседних многогранника имеют лишь одно общее ребро. Для «сетки» размера $n_1 \times n_2$ древовидная ширина вычисляется по формуле $tw(G) = \min(n_1, n_2) + 1$. При увеличении размера «сетки» ее древовидная ширина увеличивается, а число вершинной связности, наименьшая степень вершины, плотность и хроматическое число графа остаются неизменными [1]. Известно

Утверждение 2.1 [122]. *Граф имеет параметр древовидной ширины, равный k тогда и только тогда, когда он является остовным связным подграфом k-дерева.*

Такие остовные связные подграфы k-деревьев принято называть в литературе частичными k-деревьями [31, 41, 42]. Справедлива

Теорема 2.2. *Дискретная задача Вебера для n-вершинного графа с параметром древовидной ширины k может быть решена точно за время* $O(n^{k+3})$.

Доказательство. Исходя из утверждения 2.1, доказательство теоремы сводится к доказательству того факта, что ДЗВ, когда размещаемый граф является остовным подграфом k-дерева, может быть решена точно за время $O(n^{k+3})$. Для доказательства последнего необходимо и достаточно показать выполнимость трех условий. Первое условие: исходное частичное k-дерево T' может быть «насыщено» ребрами нулевого веса до k-дерева $T, T' \subseteq T$ за время, не превосходящее $O(n^{k+3})$. Второе условие: оптимальное решение π_T^* задачи $\langle T, P, F\rangle$ для k-дерева T может быть найдено за время, не превосходящее $O(n^{k+3})$. Третье условие: стоимость оптимального решения π_T^* задачи $\langle T, P, F\rangle$ должна быть равна стоимости оптимального решения $\pi_{T'}^*$ задачи $\langle T', P, F\rangle$.

Для нахождения ребер нулевого веса, которыми должно быть дополнено частичное k-дерево T' до k-дерева T необходимо решать так называемую задачу нахождения вложения частичного k-дерева в k-дерево, которая имеет в зарубежной литературе название *k-Tree Embedding Problem* [132]. Данная задача нахождения вложения в общем случае (для произвольных k) является NP-трудной, но может быть решена с помощью известного алгоритма [132] за время $O(n^{k+2})$, когда значение параметра k задано и фиксировано. Исходя из того, что $O(n^{k+2}) < O(n^{k+3})$, первое условие справедливости утверждения 2.1 выполнено. Справедливость второго условия доказывает теорема 2.1. Для доказательства справедливости последнего условия достаточно показать, что наличие новых включенных ребер нулевого веса не влияет ни на значения вычисляемых функций Беллмана $f_i(\pi_i)$, ни на элементы множества $M(\pi_i)$ для всех $\pi_i \in \Pi(X_i)$ на любом шаге i алгоритма A_T. Справедливость последнего условия доказывается тривиально. **Теорема 2.2 доказана.**

§2.2. Точный алгоритм решения задачи для k-цепи

Рассматривается задача $\langle C, P, F\rangle$, где C – есть k-цепь с n вершинами. Доказывается, что задача $\langle C, P, F\rangle$ полиномиально разрешима при фиксированном параметре k. Предлагается точный алгоритм A_{C} решения задачи $\langle C, P, F\rangle$, основанный на динамическом программировании. Предложенный алгоритм опубликован в работе [25].

2.2.1. Определение k-цепи

Приведем следующее конструктивное определение k-цепи.

Определение 1.4. [131] *Связный граф C называется k-цепью, если его построение возможно осуществить рекурсивно по правилам: полный граф из $k+1$ вершин есть k-цепь C; k-цепь с $i+1$ вершинами получается из k-цепи C с i вершинами путем добавления в нее новой вершины с номером $i+1$ и k ребер таким образом, чтобы новая вершина стала смежной со всеми вершинами, номера которых принадлежат множеству $\{(i-k)+1,\ (i-k)+2, ...,\ i\}$.*

Из определения 1.4 легко увидеть, что k-цепь является частным случаем k-дерева. Подробное исследование свойств k-цепи приведено в [42, 96, 123]. Интересно заметить, что для k-цепи также имеет место следующее определение. Пусть $N(j)$ – множество вершин графа C, смежных с вершиной j. Пусть $\varphi(C)$ – величина плотности графа C, равная размеру максимальной клики в C. Будем полагать, что на множестве вершин $V(C)$ введена нумерация и каждая вершина отождествлена с присвоенным ей номером.

Определение 1.5. *Связный граф C называется k-цепью, если на множестве его вершин можно задать такую нумерацию, что для любой вершины графа C с номером j имеет место равенство*

$$N(j) = \{(j-k), ..., (j-1), (j+1), ..., (j+k)\} \cap \{1, 2, 3, ..., n\} : k = \varphi(G) - 1.$$

На рисунке 2.3 для различных k представлены k-цепи. То есть 1-цепь пред-

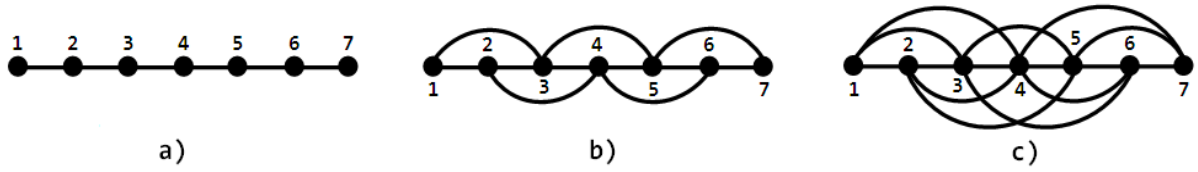

Рис. 2.3. k-цепи: a) 1-цепь; b) 2-цепь; c) 3-цепь

ставляет собой обыкновенную цепь, так как для любых j верно неравенство $|N(j)| \leq 2$. Заметим, что каждая вершина такой цепи связана не более чем с одной предшествующей вершиной, где под предшествующей вершиной понимается вершина с меньшим номером. В 2 (3)-цепи каждая вершина связана не более чем с двумя (тремя) предшествующими вершинами соответственно. Важно отметить, что такой специальный способ связи каждой вершины с предшествующими будет непосредственно использоваться в построении точного алгоритма решения ДЗВ для k-цепи.

2.2.2. Точный алгоритм размещения k-цепи

Идея алгоритма A_C решения задачи $\langle C, P, F \rangle$ заключается в следующем. На множестве $V(C)$ вершин k-цепи C задается нумерация, согласно определению 1.5. Каждая вершина графа отождествляется с присвоенным ей порядковым номером. Пусть K_i — клика размера k с множеством вершин $\{i+1, i+2, ..., i+k\} \subset V(C)$ и $\Pi(K_i) = \{\pi_i : K_i \to P\}$ – множество всех отображений вершин клики K_i в P, где $\pi_i = \{\pi_i(j) : j \in K_i\}$. Отметим, что для любых $i,\ 0 \leq i \leq n-k$ справедливо равенство $|\Pi(K_i)| = n^k$. Пусть C_i – подграф графа C, индуцированный вершинами $\{1, 2, ..., i+k\}$. Обозначим $C(\pi_i)$ – граф C_i, в котором размещение вершин его клики K_i в множестве P равно $\pi_i \in \Pi(K_i)$ (см. рис. 2.4). Множество состояний процесса ДП на шаге i будем полагать равным $\Pi(K_i)$. Значение функции Беллмана $f_i(\pi_i)$, вычисленное для некоторого состояния $\pi_i \in \Pi(K_i)$, есть стоимость оптимального размещения подграфа $C(\pi_i)$ в

множестве P.

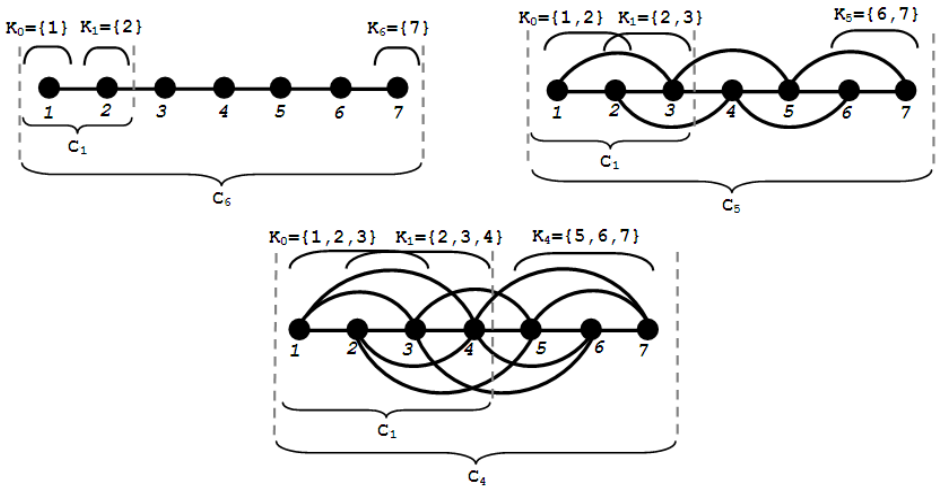

Рис. 2.4. k-клики K_i и подграфы C_i в C

Алгоритм A_C

Вход: k-цепь C; множество позиций P; функции $b(\cdot)$ и $c(\cdot)$.

Выход: отображение π^* вершин графа C в множество P.

Прямой ход.

Шаг 0. Для каждого состояния $\pi_0 \in \Pi(K_0)$ вычислить значение функции Беллмана $f_0(\pi_0)$ по формуле

$$f_0(\pi_0) = \sum_{j \in K_0} b(j, \pi_0(j)) + \sum_{j,l \in K_0} c([j,l], \pi_0(j), \pi_0(l)).$$

Шаг i, $i = 1, ..., n-k$. Для каждого состояния $\pi_i \in \Pi(K_i)$ вычислить значение функции Беллмана $f_i(\pi_i)$ по формуле

$$f_i(\pi_i) = b(i+k, \pi_i(i+k)) + \sum_{j \in K_i} c([i+k, j], \pi_i(i+k), \pi_i(j)) +$$

$$+ \min_{\pi_{i-1} \in \Pi(K_{i-1})} \{c([i+k, i], \pi_i(i+k), \pi_{i-1}(i)) + f_{i-1}(\pi_{i-1})\}, \qquad (2.5)$$

где выполняется $\pi_{i-1} \cap \pi_i = k-1$. Для всех $\pi_i \in \Pi(K_i)$ вычислить

$$M(\pi_i) = \arg \min_{\pi_{i-1} \in \Pi(K_{i-1})} \{c([i+k, i], \pi_i(i+k), \pi_{i-1}(i)) + f_{i-1}(\pi_{i-1})\},$$

где $\pi_{i-1} \cap \pi_i = k-1$.

Шаг $n-k+1$. Вычислить стоимость f^* оптимального размещения вершин графа C в P по формуле

$$f^* = \min_{\pi_{n-k} \in \Pi(K_{n-k})} f_{n-k}(\pi_{n-k}).$$

Обратный ход.

Положить $\pi^* = \varnothing$, $\pi^*_K = \pi^*_{n-k} : f_{n-k}(\pi^*_{n-k}) = f^*$ и $i = n-k$.

Шаг 1. Включить π^*_K в множество π^*. Вычислить $\pi^*_K = M(\pi^*_K)$. Положить $i = i-1$. Если $i > 1$, то повторить шаг 1, иначе **стоп**.

Конец алгоритма.

Теорема 2.3. *Алгоритм A_{C} за время $O(n^{k+2})$ находит точное решение n-вершинной задачи $\langle C, P, F\rangle$. Оценка объема требуемой оперативной памяти равна $O(n^k)$.*

Доказательство. Докажем, что размещение π^*, найденное алгоритмом A_{C}, оптимально. Обозначим $\langle C(\pi_i), P, F\rangle$ задачу оптимального размещения подграфа $C(\pi_i)$ в множестве позиций P. На шаге 0 прямого хода алгоритма, для всех $\pi_0 \in \Pi(K_0)$ величина $f_0(\pi_0)$ является минимумом задачи $\langle C(\pi_0), P, F\rangle$, поскольку вычисляется тривиально.

Докажем, что для нахождения оптимальной стоимости размещения графа $C(\pi_i),\ \pi_i \in \Pi(K_i)$ достаточно знать стоимости оптимального размещения графов $C(\pi_{i-1}),\ \pi_{i-1} \cap \pi_i = k-1$, вычисленные на предыдущем шаге $i-1$. Нетрудно заметить, что такие множества π_{i-1} и π_i должны иметь ровно $k-1$ одинаковых размещений вершин с номерами $i+1, ..., i+k-1$, причем поскольку разме-

щение $\pi_i(i+k) \in \pi_i$ фиксировано, то количество размещений $\pi_{i-1} \in \Pi(K_{i-1})$, удовлетворяющих условию $\pi_{i-1} \cap \pi_i = k-1$, равно $|P| = n$. Так как вершина $i+k$, добавляемая на шаге i, связана в подграфе $C(\pi_i)$ только с вершинами клики K_{i-1}, то, в силу того, что в одну позицию возможно разместить несколько вершин графа, для вычисления оптимальной стоимости размещения графа $C(\pi_i)$ достаточно перебрать все возможные размещения $\pi_{i-1}(i) \in P$ вершины i с целью минимизации суммы величин $c([i+k,i], \pi_i(i+k), \pi_{i-1}(i))$ и $f_{i-1}(\pi_{i-1})$. Из того факта, что для всех $\pi_{i-1} \in \Pi(K_{i-1})$ величина $f_{i-1}(\pi_{i-1})$ есть минимум задачи $\langle C(\pi_{i-1}), P, F\rangle$, естественно следует, что на шаге $i = 1, ..., n-k$ для каждого $\pi_i \in \Pi(K_i)$ величина $f_i(\pi_i)$ является минимумом задачи $\langle C(\pi_i), P, F\rangle$. Таким образом, рекуррентно решая задачи $\langle C(\pi_i), P, F\rangle,\ \pi_i \in \Pi(K_i)$, алгоритм A_C после завершения шага $n-k$, перебирая все возможные способы размещения клики K_{n-k}, вычисляет оптимальную стоимость f^* размещения исходного графа C. На обратном ходе алгоритм, зная хранимые в памяти значения функции $M(\cdot)$, рекуррентно строит оптимальное размещение π^*.

Докажем оценки объема требуемой оперативной памяти и времени работы алгоритма A_C. Для решения задачи $\langle C(\pi_i), P, F\rangle$ необходимо $O(n)$ операций, поскольку для вычисления значения функций $f_i(\pi_i)$ и $M(\pi_i)$ достаточно перебрать все возможные размещения $\pi_{i-1}(i) \in P$. Отсюда, на шаге $i = 1, ..., n-k$ требуется $O(n^{k+1})$ операций, и трудоемкость прямого хода не превосходит $O(n^{k+2})$. Так как трудоемкость обратного хода равна $O(n)$, то *оценка времени работы* алгоритма A_C равна $O(n^{k+2})$. Положим, что для хранения одного значения функции $f_i(\cdot)$ требуется $O(1)$ памяти. Также положим, что для хранения одного значения $M(\cdot)$ требуется $O(1)$ памяти, поскольку, в данном случае, нам достаточно хранить лишь номер соответствующей ячейки в массиве Π_{i-1}. На шаге $i = 2, ..., n-k$ требуется $O(n^k)$ памяти, так как для хранения каждого массива $f_i(\cdot)$, $f_{i-1}(\cdot)$ и $M(\cdot)$ необходимо ровно $O(n^k)$ памяти. Поскольку для вычисления значений функции $f_i(\cdot)$ на шаге i требуется хранить лишь

единственный массив значений $f_{i-1}(\cdot)$, сформированный на предыдущем шаге, то *оценкой требуемой оперативной памяти* алгоритма A_C не превосходит $O(n^k)$. **Теорема 2.3 доказана.**

Стоит заметить, что при любом фиксированном значении входного параметра k сложность алгоритма A_C, так же, как и алгоритма A_T, полиномиальна и ограничена полиномом порядка степени $k+2$. Отметим, что, благодаря учету специфики задачи, алгоритм A_C имеет существенно лучшие оценки трудоемкости и требуемого объема памяти, чем более общий алгоритм A_T.

2.2.3. Обобщение алгоритма для k-цепи на класс графов с ограниченной ленточной шириной

Покажем, что факт существования точного алгоритма решения ДЗВ для k-цепи с трудоемкостью $O(n^{k+2})$ непосредственно влечет факт того, что любая ДЗВ для графа с параметром *ленточной ширины* равным k также может быть разрешена точно за время, равное $O(n^{k+2})$.

Приведем известное определение ленточной ширины графа. Нумерацией вершин графа G будем называть инъективное отображение $\varphi : V(G) \to \{1, 2, ..., n\}$, где $|V(G)| = n$.

Определение 1.6 [105]. *Параметр $bw(G)$ ленточной ширины (bandwidth) графа G – есть число, равное наименьшей ленточной ширине всех нумераций вершин графа, где ленточная ширина некоторой нумерации φ равна*

$$\max_{i,j \in V(G)} \{|\varphi(i) - \varphi(j)| : [i, j] \in E(G)\}.$$

Классом графов с ограниченной ленточной шириной будем называть множество всех графов, для которых параметр ленточной ширины меньше либо равен k, где k – положительная константа, не зависящая от n. Класс графов с ограниченной ленточной шириной был исследован множеством авторов [57, 58, 61].

Параметр ленточной ширины графа, так же как и параметр древовидной ширины играет важную роль в современной теории графов, поскольку некоторые задачи, которые NP-трудны для произвольных графов, могут быть решены точно за полиномиальное время на классах графов, для которых такой параметр фиксирован [58]. Известно

Утверждение 2.2 [133]. *Граф имеет параметр ленточной ширины, равный k тогда и только тогда, когда он является остовным связным подграфом k-цепи.*

По аналогии с частичными k-деревьями, такие остовные подграфы принято называть частичными k-цепями. Справедлива

Теорема 2.4. *Дискретная задача Вебера для n-вершинного графа с параметром ленточной ширины k может быть решена точно за время $O(n^{k+2})$.*

Доказательство. Исходя из утверждения 2.2, доказательство теоремы 2.4, сводится к доказательству того факта, что ДЗВ, когда размещаемый граф является остовным подграфом k-цепи, всегда может быть разрешена точно за время, не превосходящее $O(n^{k+2})$. Для доказательства данного факта, так же как и в утверждении 2.1, необходимо и достаточно показать выполнимость трех условий. Первое условие: исходная частичная k-цепь C' может быть дополнена ребрами нулевого веса до k-цепи $C, C' \subseteq C$ за время, не превосходящее $O(n^{k+2})$. Второе условие: оптимальное решение π^*_C задачи $\langle C, P, F\rangle$ для k-цепи C может быть найдено за время, не превосходящее $O(n^{k+2})$. Третье условие: стоимость оптимального решения π^*_C задачи $\langle C, P, F\rangle$ должна быть равна стоимости оптимального решения $\pi^*_{C'}$ задачи $\langle C', P, F\rangle$.

В [92] доказано, что множество ребер нулевого веса, которыми должна быть дополнена исходная частичная k-цепь C' до k-цепи, может быть вычислено за время $O(n^{k+2})$. Исходя из этого, первое условие справедливости утверждения 2.2 выполнено. Справедливость второго условия доказывает теорема 2.2. Для доказательства справедливости третьего условия достаточно показать, что на-

личие новых, включенных в T, ребер нулевого веса не влияет ни на значение функции Беллмана $f_i(\pi_i)$, ни на значения функции $M(\pi_i)$ для всех $\pi_i \in \Pi(X_i)$ на любом шаге i алгоритма A_C. Справедливость последнего условия доказывается тривиально. **Теорема 2.4 доказана.**

§2.3. Точный алгоритм решения задачи для простого цикла

Рассматривается задача $\langle S, P, F\rangle$, где S – есть простой цикл. Предлагается точный алгоритм A_S решения задачи $\langle S, P, F\rangle$, основанный на динамическом программировании. Предложенный алгоритм имеет значительно лучшую оценку объема требуемой оперативной памяти, чем известный алгоритм Ф. Малучелли для последовательно-параллельного графа, предложенный в работе [110]. Результат опубликован в [24].

Введем следующие обозначения. Пусть $s,\ s \in V(S)$ – произвольная вершина цикла S, и T – подграф графа S, индуцированный множеством вершин $V(S) \setminus \{s\}$. Пусть $N, N = |V(T)|$ – мощность множества вершин цепи T и $i^* \in V(T)$ – висячая вершина цепи T. Выбор вершины i^* в качестве корня дерева T индуцирует на множестве вершин $V(T)$ отношение частичного порядка

$$L = \{(i,j) : i,j \in V(T), j \text{ принадлежит цепи в } T \text{ между } i \text{ и } i^*\}. \qquad (2.6)$$

В дальнейшем будем считать, что $I = \{1, ..., N\}$ и удовлетворяет условию $(l,m) \in L \Rightarrow l < m$, где вершины $l, m \in V(T)$. Каждую вершину цепи T будем отождествлять с присвоенным ей порядковым номером. Обозначим S_i – подграф графа S, индуцированный множеством вершин $s, 1, 2, ..., i-1, i$. На рисунке 2.5 представлены цепь T и подграф S_i в цикле S.

Положим, что $S(\pi(s)),\ \pi(s) \in P$ – есть граф S, в котором вершина s размещена в позицию $\pi(s)$. Исходную задачу $\langle S, P, F\rangle$ разобьем на ряд подзадач

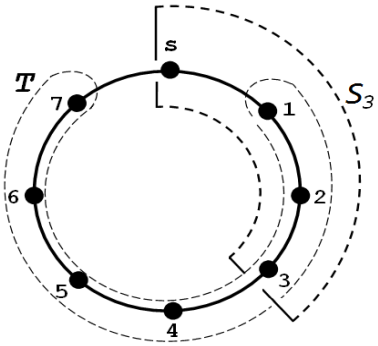

Рис. 2.5. Цепь T и подграф S_3 в простом цикле S

$\langle S(\pi(s)), P, F\rangle$ для каждого $\pi(s) \in P$. Обозначим $P = \{\pi(i) : \pi(i) \in P\}$ – множество состояний процесса ДП на шаге i решения подзадачи $\langle S(\pi(s)), P, F\rangle$, где под состоянием $\pi(i) \in P$ понимается способ размещения вершины i подграфа S_i в множестве позиций P. Значение функции Беллмана $f_i(\pi(i))$, вычисленное на шаге i решения подзадачи $\langle S(\pi(s)), P, F\rangle$ для некоторого состояния $\pi(i)$, есть стоимость оптимального размещения подграфа S_i в множестве позиций P, когда размещение вершин s и i в множестве P равно $\pi(s)$ и $\pi(i)$ соответственно.

АЛГОРИТМ A_S

Вход: простой цикл S; множество P; функции $b(\cdot)$ и $c(\cdot)$.

Выход: оптимальное отображение π^* вершин графа S в P.

Этап 0. Выбрать произвольную вершину $s \in V(S)$. Определить подграф T и на множестве его вершин задать отношение порядка (2.6).

Этап 1. Для каждого $\pi(s) \in P$ решить подзадачу $\langle S(\pi(s)), P, F\rangle$.

Прямой ход.

Шаг 1. Для каждого состояния $\pi(1) \in P$ вычислить значение функции Беллмана по формуле

$$f_1(\pi(1)) = b(1, \pi(1)) + c([1, s], \pi(1), \pi(s)) + b(s, \pi(s)).$$

Для каждого $\pi(1) \in P$ положить $M(\pi_1) = \pi(s)$.

Шаг i, $i = 2, ..., N-1$. Для каждого состояния $\pi(i) \in P$ вычислить значение функции Беллмана

$$f_i(\pi(i)) = b(i, \pi(i)) + \min_{\pi(i-1)\in P}\{c([i, i-1], \pi(i), \pi(i-1)) + f_{i-1}(\pi(i-1))\}. \quad (2.7)$$

Для каждого $\pi(i) \in P$ вычислить

$$M(\pi_i) = \arg\min_{\pi(i-1)\in P}\{c([i, i-1], \pi(i), \pi(i-1)) + f_{i-1}(\pi(i-1))\}.$$

Шаг N. Для каждого состояния $\pi(N) \in P$ вычислить значение функции Беллмана по формуле

$$f_N(\pi(N)) = b(N, \pi(N)) +$$

$$+ \min_{\pi(N-1)\in P}\{c([N, N-1], \pi(N), \pi(N-1)) + f_{N-1}(\pi(N-1)) + c([N, s], \pi(N), \pi(s))\}.$$

Для каждого $\pi(N) \in P$ вычислить

$$M(\pi_N) = \arg\min_{\pi(N-1)\in P}\{c([N, N-1], \pi(N), \pi(N-1)) +$$

$$+ f_{N-1}(\pi(N-1)) + c([N, s], \pi(N), \pi(s))\}.$$

Шаг $N+1$. Вычислить стоимость $f^*_{\pi(s)}$ оптимального решения задачи $\langle S(\pi(s)), P, F\rangle$ по формуле

$$f^*_{\pi(s)} = \min_{\pi(N)\in P}\{f_N(\pi(N))\}. \quad (2.8)$$

Обратный ход.

Положить $\pi^*_{\pi(s)} = \varnothing$, $i = N$ и $\pi(N)^* = \arg\min_{\pi(N)\in P}\{f_N(\pi(N))\}$.

Шаг 1. Включить $\pi(i)^*$ в множество $\pi^*_{\pi(s)}$. Вычислить $\pi^*_{i-1} = M(\pi(i)^*)$. Положить $i = i - 1$. Если $i > 1$, то повторить шаг 1, иначе повторить этап 1 для следующего $\pi(s)$.

Этап 2. После того, как оптимальное решение подзадачи $\langle S(\pi(s)), P, F\rangle$ для каждого $\pi(s) \in P$ найдено, определить оптимальное решение π^* исходной задачи $\langle S, P, F\rangle$:

$$\pi^* = \pi^*_{\pi(s)^*} : \pi(s)^* = \arg \min_{\pi(s)\in P}\{f^*_{\pi(s)}\}. \tag{2.9}$$

Стоп.

Конец алгоритма.

Теорема 2.5. *Алгоритм A_S за время $O(n^4)$ находит точное решение n-вершинной задачи $\langle S, P, F\rangle$. Оценка требуемого объема оперативной памяти равна $O(n^2)$.*

Доказательство. Докажем, что подзадача $\langle S(\pi(s)), P, F\rangle$ при каждом фиксированном размещении $\pi(s) \in P$ решается алгоритмом A_S оптимально. Очевидно, на шаге 2 для любых $\pi(s), \pi(2) \in P$ алгоритм находит оптимальное размещение вершины 1 относительно заданного размещения $\pi(s), \pi(2)$ вершин s и 2, причем, в данном случае, это достигается путем полного перебора всех способов размещения $\pi(1)$ вершины 1 в множестве позиций P. На шаге 3 для любых $\pi(s), \pi(3) \in V$ алгоритм находит оптимальное размещение вершин 1 и 2 относительно заданного размещения $\pi(s), \pi(3)$ вершин s и 3, поскольку для вершины 1 оптимальное размещение определено на предыдущем шаге 1 для каждого заданного размещения $\pi(2)$ вершины 2, а оптимальное размещение вершины 2 согласно формуле (2.7) находится перебором способов размещения $\pi(2)$ в множестве позиций P с целью минимизации стоимости размещения подграфа T_3, когда размещение вершин s и 3 в P равно $\pi(s)$ и $\pi(3)$ соответственно.

Так, на последующих шагах $i = 4, 5, ..., N$ для любых $\pi(s), \pi(i) \in P$, алгоритм находит оптимальное размещение вершин $1, 2, ..., i-1$ относительно заданного размещения $\pi(s), \pi(i)$ вершин s и i, поскольку для вершин $1, 2, ..., i-2$ оптимальное размещение определено на предыдущем шаге $i-1$ для каждого

заданного размещения $\pi(i-1)$ вершины $i-1$, а оптимальное размещение вершины $i-1$ согласно формуле (2.7) находится перебором способов размещения $\pi(i-1)$ в множестве позиций P с целью минимизации стоимости размещения подграфа T_i, когда размещение вершин s и i равно $\pi(s)$ и $\pi(i)$.

Таким образом, рекуррентно определяя оптимальное размещение вершин подграфа T_i на шагах $i = 2, 3, ..., N$, алгоритм на шаге $N+1$, согласно формуле (2.8), зная оптимальное размещение вершин $1, 2, ..., N-1$ относительно любых заданных размещений $\pi(s)$ и $\pi(N)$, находит оптимальную стоимость размещение графа $S(\pi(s))$. Поскольку размещение графа $S(\pi(s))$ для любых $\pi(s) \in P$ найдено оптимально, то алгоритм на этапе 2, осуществив полный перебор способов размещения $\pi(s)$ вершины s, согласно формуле (2.9), находит оптимальное решение исходной задачи $\langle S, P, F \rangle$.

Определим оценку числа операций, требуемых для решения одной подзадачи $\langle S(\pi(s)), P, F \rangle$. На шаге 1 необходимо выполнить $O(n)$ операций, так как $O(n)$ операций требуется для вычисления значений функции Беллмана $f_1(\cdot)$ и $O(n)$ операций требуется для вычисления значений функции $M(\cdot)$. На каждом из последующих шагов $i = 2, 3, ..., N$ требуется выполнить $O(n^2)$ операций, поскольку $O(n^2)$ операций необходимо для вычисления значений функции Беллмана $f_i(\cdot)$, и $O(n)$ операций необходимо для вычисления значений функции $M(\cdot)$. На шаге $N+1$ требуется выполнить $O(n)$ операций, так как $O(n)$ операций необходимо для вычисления значения функции $f^*_{\pi(s)}$ стоимости оптимального размещения графа $S(\pi(s))$. В соответствии с этим, для решения одной подзадачи $\langle S(\pi(s)), P, F \rangle$ требуется $O(n^3)$ времени вычислений, где

$$O(n^3) = O(n) + O(n^2) \cdot (n-2) + O(n).$$

Исходя из того, что количество решаемых подзадач $\langle S(\pi(s)), P, F \rangle$ равно n, оценка *времени работы* алгоритма A_S равна $O(n^4)$.

Определим оценку оценкой требуемой оперативной памяти алгоритма. Положим, что для хранения одного значения функций $f_i(\cdot)$ и $M(\cdot)$ требуется $O(1)$ памяти. При решении любой подзадачи $\langle S(\pi(s)), P, F\rangle$ на шаге i требуется $O(n)$ памяти, так как $O(n)$ памяти требуется для хранения массивов $f_i(\cdot)$ и $M(\cdot)$, а $O(n)$ памяти необходимо для хранения значений $f_{i-1}(\cdot)$ вычисленных на предыдущем шаге $i-1$, причем необходимость хранения в памяти на шаге i массивов $f_j(\cdot)$, вычисленных на всех предшествующих шагах $j = i-2, i-3, ...$ отпадает. Так как массивы $M(\cdot)$ вычисленные на всех шагах необходимо хранить в памяти для осуществления обратного хода, то решение подзадачи $\langle S(\pi(s)), P, F\rangle$ требует $O(n^2)$ памяти. Поскольку подзадачи $\langle S(\pi(s)), P, F\rangle$ решаются независимо друг от друга, то оценка *объема требуемой оперативной памяти* алгоритма A_S равна $O(n^2)$. **Теорема 2.5 доказана.**

§2.4. Выводы по второй главе

Во второй главе рассмотрены три подкласса ДЗВ, когда структура связей между размещаемыми объектами задана k-деревом, k-цепью, являющейся частным случаем k-дерева, и простым циклом.

Для частного случая ДЗВ, когда размещаемый граф представляет собой k-дерево, предложен полиномиальный при фиксированном k точный алгоритм решения A_T, основанный на динамическом программировании по дереву декомпозиции. Оценка времени работы алгоритма – $O(n^{k+3})$, оценка требуемого объема оперативной памяти – $O(n^{k+3})$, где n – число вершин графа. Используя предложенный точный алгоритм для k-дерева, было доказано, что ДЗВ полиномиально разрешима на классе графов, для которых параметр древовидной ширины (treewidth) фиксирован и равен k. Последний полученный результат обобщает известный в литературе результат Ф. Малучелли [110] о полиномиальной разрешимости ДЗВ для последовательно-параллельных графов.

Для частного случая ДЗВ, когда размещаемый граф является k-цепью, предложен точный полиномиальный при фиксированном k алгоритм решения A_C, основанный на динамическом программировании. Оценка времени работы алгоритма – $O(n^{k+2})$, оценка требуемого объема оперативной памяти – $O(n^k)$, где n – число вершин графа. Благодаря учету специфики задачи, данный алгоритм имеет лучшие оценки времени работы и объема требуемой оперативной памяти, чем более общий алгоритм для k-дерева. Используя, предложенный точный алгоритм для k-цепи было доказано, что ДЗВ полиномиально разрешима на классе графов, для которых параметр ленточной ширины (bandwidth) фиксирован и равен k.

Для частного случая ДЗВ, когда размещаемый граф является простым циклом, предложен точный полиномиальный алгоритм решения A_S. Оценка времени работы алгоритма – $O(n^4)$, оценка требуемого объема оперативной памяти – $O(n^2)$. Предложенный алгоритм, благодаря учету специфики задачи, имеет значительно лучшую оценку объема требуемой оперативной памяти, чем более общий алгоритм Ф. Малучелли для последовательно-параллельного графа, предложенный в [110], хотя оценки трудоемкостей таких алгоритмов одинаковы.

Глава 3.

Алгоритмы с оценками точности

Данная глава посвящена алгоритмам с оценками точности решения ДЗВ для графов как специальной, так и произвольной структуры. В разделе 3.1 предложен полиномиальный приближенный алгоритм Арх, находящий за время $O(n^3)$ решение ДЗВ для случая произвольного размещаемого графа. Доказана апостериорная оценка точности алгоритма Арх. Выявлены параметры задачи, влияющие на величину апостериорной оценки точности алгоритма Арх. Найдены подклассы задачи, на которых алгоритм является 2-приближенным и ассимптотически точным. В разделе 3.2 построена метаэвристика с апостериорной оценкой точности решения ДЗВ для графов произвольной структуры, сочетающая в себе идеи предложенного приближенного алгоритма Арх и генетического алгоритма.

§3.1. Приближенный алгоритм

Обозначим $F(X, \pi)$ стоимость размещения графа X, когда размещение его вершин в множестве P равно $\pi \in \Pi$. Обозначим H – суграф графа G, то есть $V(H) = V(G)$ и $E(H) \subseteq E(G)$. Пусть $\Delta(H, \pi) = F(G, \pi) - F(H, \pi)$ – стоимость размещения ребер графа G, не вошедших в суграф H, когда размещение их концевых вершин принадлежит π. Справедлива

Теорема 3.1. *Если H – суграф графа G и π_H – оптимальное размещение H в P, то стоимость решения π_H задачи $\langle G, P, F\rangle$ превосходит стоимость*

оптимального решения π_G задачи $\langle G, P, F\rangle$ не более чем в ε раз, где

$$\varepsilon = 1 + \frac{\Delta(H, \pi_H)}{F(H, \pi_H)}. \tag{3.1}$$

Доказательство. Поскольку решение π_H является оптимальным решением задачи $\langle H, P, F\rangle$, то

$$(\forall \pi \in \Pi) \qquad F(H, \pi_H) \leq F(H, \pi). \tag{3.2}$$

Обозначим π_G оптимальное решение задачи $\langle G, P, F\rangle$. Исходя из (3.2) естественно следует $F(H, \pi_H) \leq F(H, \pi_G)$. Так как величина $\Delta(\cdot)$ всегда ≥ 0, то справедливо условие

$$F(H, \pi_H) \leq F(H, \pi_G) + \Delta(H, \pi_G) = F(G, \pi_G), \tag{3.3}$$

что, очевидно, определяет значение нижней границы задачи $\langle G, P, F\rangle$. Поскольку $F(\cdot) \geq 0$, то справедливо неравенство

$$F(G, \pi_H) \cdot F(H, \pi_H) \leq F(G, \pi_H) \cdot F(G, \pi_G),$$

откуда следует, что для решения π_H задачи $\langle G, P, F\rangle$ справедлива оценка

$$F(G, \pi_H) \leq \frac{F(G, \pi_H)}{F(H, \pi_H)} \cdot F(G, \pi_G) = \left(1 + \frac{\Delta(H, \pi_H)}{F(H, \pi_H)}\right) \cdot F(G, \pi_G).$$

Теорема 3.1 доказана.

Иными словами величина оценки ε определяется отношением стоимости размещения ребер графа G, не вошедших в суграф H, когда размещение их концевых вершин принадлежит π_H, к стоимости оптимального размещения вершин и ребер такого остовного подграфа H. Важно заметить, что, если при решении некоторой индивидуальной задачи $\langle G, P, F\rangle$, величина стоимости размещения

ребер $\Delta(H, \pi_H)$ равна 0, то это автоматически доказывает оптимальность найденного решения π_H такой индивидуальной задачи.

Оценка (3.1) позволяет построить эффективный алгоритм, находящий допустимое приближенное решение задачи $\langle G, P, F\rangle$ и вычисляющий оценку точности найденного решения. Для этого достаточно уметь строить «аппроксимирующий» суграф H графа G, а также вычислять оптимальное размещение вершин графа H за полиномиальное время. В качестве такого «аппроксимирующего» суграфа в данной работе будем использовать дерево, поскольку алгоритмы построения остова общеизвестны [65, 119], а решение задачи ДЗВ для дерева может быть найдено за полиномиальное время [44, 116].

Рассмотрим влияние «аппроксимирующего» суграфа H на оценку точности. Поскольку

$$\frac{\Delta(H, \pi_H)}{F(H, \pi_H)} = \frac{\sum_{[i,j]\in E(G)\setminus E(H)} w(i,j)\cdot d(\pi_H(i), \pi_H(j))}{\sum_{i\in V(G)} b(i, \pi_H(i)) + \sum_{[i,j]\in E(H)} w(i,j)\cdot d(\pi_H(i), \pi_H(j))}, \qquad (3.4)$$

то остовной граф H следует строить, максимизируя суммарный вес включенных ребер. Действительно, в выражении (3.4) числитель не увеличивается, а знаменатель не уменьшается с увеличением веса графа H. Ниже приведено описание приближенного алгоритма Арх решения задачи $\langle G, P, F\rangle$.

Алгоритм Арх

Вход: простой взвешенный граф G; множество P; функции $b(\cdot)$ и $c(\cdot)$.

Выход: отображение π_G вершин графа G в множество P; оценка точности ε.

Шаг 1. Найти остовное дерево T максимального веса в графе G.

Шаг 2. Вычислить оптимальное размещение π_T вершин дерева T в P.

Шаг 3. Положить $\pi_G = \pi_T$ и $\varepsilon = 1 + \Delta(T, \pi_T)/F(T, \pi_T)$.

Стоп.

Конец алгоритма.

Нетрудно заметить, что для нахождения остовного дерева T *максималь-*

ного веса на шаге 1 достаточно умножить веса ребер графа G на -1, а затем применить известный алгоритм, строящий дерево *минимального* веса. Вычислительная сложность алгоритма Арх решения задачи $\langle G, P, F\rangle$ равна $O(n^3)$, поскольку трудоемкость нахождения остовного дерева T максимального веса, например с помощью алгоритма Краскала [65], равна $O(|E(G)|\log|E(G)|)$, а вычисление оптимального размещения дерева T, например, с помощью алгоритма из работы [116], требует $O(n^3)$ операций. Очевидно, что $\varepsilon \neq const$ и зависит от входных данных каждой индивидуальной задачи. Поскольку значение ε становится известным только после проведения вычислений алгоритма, то величина ε называется относительной *апостериорной* оценкой точности алгоритма Арх.

Для определенных классов задач может быть определена *априорная* оценка точности алгоритма Арх. Приведем определение α-приближенного алгоритма. В соответствии с [5] алгоритм A решения комбинаторной оптимизационной задачи I называется *α-приближенным*, если при любых входных параметрах задачи I он за полиномиальное время находит допустимое решение, вес которого отличается от веса оптимального решения данной индивидуальной задачи I не более чем в α раз. Справедлива

Теорема 3.2. *Для всех индивидуальных задач $\langle G, P, F\rangle$ с функциями b, w, d такими, что имеет место неравенство*

$$\max_{t,u\in P}\{d(t,u)\}\cdot\sum_{[i,j]\in E(G)} w(i,j) \leq \sum_{i\in V(G)}\min_{t\in P}\{b(i,t)\}, \tag{3.5}$$

алгоритм Арх *является 2-приближенным алгоритмом.*

Доказательство. Необходимым и достаточным условием неравенства $\varepsilon \leq 2$ является неравенство $\Delta(T,\pi_T) \leq F(T,\pi_T)$. Поэтому достаточно доказать, что для любых $\pi_T \in \Pi$ выполнение условия (3.5) влечет справедливость неравенства $\Delta(T,\pi_T) \leq F(T,\pi_T)$. Данное неравенство в терминах функций b и $c = d\cdot w$

имеет вид

$$\sum_{[i,j]\in E(G)} w(i,j)d(\pi_T(i),\pi_T(j)) \le \sum_{i\in V(G)} b(i,\pi_T(i))+$$

$$+2\sum_{[i,j]\in E(T)} w(i,j)d(\pi_T(i),\pi_T(j)). \qquad (3.6)$$

Поскольку для любых $\pi \in \Pi$ справедливы неравенства

$$\sum_{[i,j]\in E(G)} w(i,j)d(\pi(i),\pi(j)) \le \max_{t,u\in P}\{d(t,u)\}\cdot \sum_{[i,j]\in E(G)} w(i,j)$$

и

$$\sum_{i\in V(G)} b(i,\pi(i)) + 2\sum_{[i,j]\in E(T)} w(i,j)d(\pi(i),\pi(j)) \ge \sum_{i\in V(G)} \min_{t\in P}\{b(i,t)\},$$

то, справедливость неравенства (3.6) является очевидным следствием условия (3.5) теоремы. **Теорема 3.2 доказана.**

Условие (3.5) имеет следующий смысл: минимальная суммарная стоимость размещения вершин графа G больше либо равна максимальной возможной стоимости размещения его ребер. Заметим, как показывал вычислительный эксперимент на случайных данных, такое условие очень часто возникает, когда среднее значение величин $b(\cdot)$ намного больше, чем произведение средних значений величин $d(\cdot)$ и $w(\cdot)$.

Отметим, что аналогичным образом для некоторой констаты $1 < \alpha < 2$ можно определить класс индивидуальных задач $\langle G, P, F\rangle$, на которых АРХ будет α-приближенным алгоритмом. В данном случае необходимость выполнения неравенства $\Delta(T,\pi_T) \le F(T,\pi_T)$ заменится на требование верности условия $\Delta(T,\pi_T) \le (1-\alpha)F(T,\pi_T)$. Также легко заметить тот факт, что при $b \to \infty$ и $w, d \to 0$ величина оценки $\varepsilon \to 1$. Экономическая интерпретация последнего факта следующая: чем больше стоимость постройки предприятий и чем меньше величина планируемого грузового потока между предприятиями и расстояния

между потенциальными позициями размещения, тем точнее алгоритм Арх решает задачу.

Рассмотрим ДЗВ $\langle S, P, F\rangle$, где S – простой взвешенный цикл. Справедлива

Теорема 3.3. *Для задачи $\langle S, P, F\rangle$ с n вершинами алгоритм* Арх *за время $O(n^3)$ находит приближенное решение, стоимость которого не более чем в 2 раза превосходит оптимум.*

Доказательство. Трудоемкость алгоритма Арх решения задачи $\langle S, P, F\rangle$ доказывается аналогичным образом, как для задачи $\langle G, P, F\rangle$.

Пусть T – остовное дерево (в данном случае цепь) максимального веса в S. Обозначим $[i^*, j^*]$ ребро наименьшего веса в цикле S. Исходя из неравенства (3.1), справедлива следующая оценка точности алгоритма Арх для задачи $\langle S, P, F\rangle$:

$$F(S, \pi_T) \le \left(1 + \frac{c([i^*, j^*], \pi_T(i^*), \pi_T(j^*))}{F(T, \pi_T)}\right) \cdot F(S, \pi_S), \tag{3.7}$$

В терминах функций w, d и b имеем

$$\frac{c([i^*, j^*], \pi_T(i^*), \pi_T(j^*))}{F(T, \pi_T)} = \left(\frac{\sum_{[i,j]\in E(T)} w(i,j) d(\pi_T(i), \pi_T(j))}{w(i^*, j^*) d(\pi_T(i^*), \pi_T(j^*))} + \right.$$

$$\left. + \frac{\sum_{i \in V(S)} b(i, \pi(i))}{w(i^*, j^*) d(\pi_T(i^*), \pi_T(j^*))}\right)^{-1}. \tag{3.8}$$

Поскольку функция d является метрикой, то

$$d(\pi_T(i^*), \pi_T(j^*)) \le \sum_{[i,j]\in E(T)} d(\pi_T(i), \pi_T(j)). \tag{3.9}$$

Пусть

$$(\forall [i, j] \in E(T)) \qquad \lambda(i, j) = \frac{w(i, j)}{w(i^*, j^*)} \ge 1. \tag{3.10}$$

Исходя из неравенств (3.9-3.10) имеем

$$\sum_{[i,j]\in E(T)} \frac{w(i,j)}{w(i^*,j^*)} d(\pi_T(i),\pi_T(j)) = \sum_{[i,j]\in E(T)} \lambda(i,j) d(\pi_T(i),\pi_T(j)) \geq d(\pi_T(i^*),\pi_T(j^*)).$$

Поскольку функции w, d и b принимают только положительные значения, то легко заметить, что величина левого слагаемого в (3.8) всегда будет не меньше 1, а величина правого слагаемого формулы (3.8) всегда неотрицательна. Следовательно, для соотношения (3.8) выполняется неравенство

$$\frac{c([i^*,j^*],\pi_T(i^*),\pi_T(j^*))}{F(T,\pi_T)} \leq 1,$$

что, в свою очередь, влечет справедливость априорной оценки точности

$$1 + \frac{c([i^*,j^*],\pi(i^*),\pi(j^*))}{F(T,\pi_T)} \leq 2.$$

Теорема 3.3 доказана.

Практический интерес представляет поведение погрешности с ростом размерности задачи. С этой целью введено понятие асимптотической точности [5]. Пусть K некоторый класс задач минимизации. Для задачи $I \in K$ обозначим через $OPT(I)$ оптимальное значение целевой функции, а через $A(I)$ – значение, найденное приближенным алгоритмом A. Пусть $K_n \subset K$ – подмножество задач размерности n. Алгоритм A называется *асимптотически точным*, если

$$(\forall I \in K_n) \qquad A(I) \leq (1 + e_n) \cdot OPT(I),$$

где $e_n \to 0$ при $n \to \infty$

Теорема 3.4. *Алгоритм* Арх *решения задачи* $\langle S, P, F\rangle$ *является асимптотически точным.*

Доказательство. Исходя из теоремы 3.1, справедлива следующая оценка

точности алгоритма Арх для задачи $\langle S, P, F\rangle$:

$$F(S, \pi_T) \leq \left(1 + \frac{c([i^*, j^*], \pi_T(i^*), \pi_T(j^*))}{F(T, \pi_T)}\right) \cdot F(S, \pi_S), \tag{3.11}$$

Запишем второе слагаемое в терминах функций w, d и b:

$$\left(\frac{\sum_{[i,j]\in E(T)} w(i,j)d(\pi_T(i), \pi_T(j))}{w(i^*, j^*)d(\pi_T(i^*), \pi_T(j^*))} + \frac{\sum_{i\in V(S)} b(i, \pi(i))}{w(i^*, j^*)d(\pi_T(i^*), \pi_T(j^*))}\right)^{-1}. \tag{3.12}$$

Получаем оценки слагаемых в соотношении (3.12) при $n \to \infty$:

$$\frac{\sum_{[i,j]\in E(T)} w(i,j)d(\pi_T(i), \pi_T(j))}{w(i^*, j^*)d(\pi_T(i^*), \pi_T(j^*))} \geq n \cdot \frac{\sum_{[i,j]\in E(T)} d(\pi_T(i), \pi_T(j))}{d(\pi_T(i^*), \pi_T(j^*))} \xrightarrow[n\to\infty]{} \infty,$$

$$\frac{\sum_{i\in V(S)} b(i, \pi(i))}{w(i^*, j^*)d(\pi_T(i^*), \pi_T(j^*))} \geq n \cdot \frac{\min_{i\in V(S)}\{b(i, \pi_T(i))\}}{w(i^*, j^*)d(\pi_T(i^*), \pi_T(j^*))} \xrightarrow[n\to\infty]{} \infty$$

Следовательно, в соотношении (3.11)

$$\frac{c([i^*, j^*], \pi_T(i^*), \pi_T(j^*))}{F(T, \pi_T)} \xrightarrow[n\to\infty]{} 0,$$

из чего следует, что оценка точности $\varepsilon \to 1$ при $n \to \infty$.

Теорема 3.4 доказана.

§3.2. Модификации приближенного алгоритма

Использование дерева в алгоритме Арх в качестве «аппроксимирующего» остовного подграфа выгодно по двум причинам: во-первых, построить остов можно за полиномиальное время с помощью известных алгоритмов; во-вторых, задача размещения дерева может быть сравнительно быстро разрешена известным алгоритмом. С другой стороны, имеется существенный недостаток такого подхода – малое число ребер в дереве, так как, в соответствии с условием (3.4), чем больше вес ребер «аппроксимирующего» основного подграфа, тем

меньше апостериорная оценка точности решения. Исходя из этого, предлагается использовать в качестве «аппроксимирующего» остовного подграфа известное обобщение дерева – частичное k-дерево, число ребер которого ограничено $nk - k(k+1)/2$ тогда, как число ребер в дереве равно $n-1$.

Использование частичного k-дерева в качестве «аппроксимирующего» остовного подграфа в приближенном алгоритме требует выполнения двух условий: первое – наличие полиномиального алгоритма построения такого подграфа в графе G; второе – существование полиномиального алгоритма размещения графа такого вида. Последнее условие выполнимо, поскольку в главе 1.1 предложен точный алгоритм A_T решения ДЗВ для k-дерева с трудоемкостью $O(n^{k+3})$, что, согласно утверждению 1.1, дает возможность эффективного решения задачи для частичного k-дерева. Построение же максимального частичного k-дерева в графе требует решения задачи, известной в зарубежной литературе как *Maximal Spanning k-tree Problem* (далее MSkT), которая является обобщением классической задачи нахождения минимального остовного дерева [40]. Формулировка задачи следующая.

Пусть G – простой взвешенный граф без петель и кратных ребер, причем для каждого ребра $[i,j] \in E(G)$ задан его вес $w(i,j) \geq 0$. Задана положительная целочисленная константа k. Обозначим $T(G)$ – множество всех возможных остовных частичных k-деревьев в графе G. Пусть $w(T)$ суммарный вес ребер остовного частичного k-дерева $T \in T(G)$. Требуется найти остовное частичное k-дерево T^* максимального веса в полном взвешенном графе G, то есть $T^* = \arg\max_{T \in T(G)}\{w(T)\}$.

Известно, что задача MSkT является NP-трудной при $k \geq 2$, как для полного графа, так и для графов с ограниченной степенью вершин, расщепляемых и планарных графов [40, 51]. В [40] доказано, что мощность множества $T(G)$ допустимых решений задачи MSkT равна $|V|!(|V|+k-1)!/k!$, исходя из чего трудоемкость полного перебора составляет порядка $O(|V|^{2|V|}/k^k)$ операций. Из-

вестен точный неполиномиальный алгоритм, основанный на динамическом программировании, имеющий значительно меньшую трудоемкость по сравнению с полным перебором, равную $O(|V|^{k+1}3^{|V|})$ операций [40]. В работах [37–39] предложены эффективные эвристические и приближенные алгоритмы для решения задачи MS2T (то есть при $k = 2$).

Для полноты изложения, в качестве примера, приведем жадную эвристику Greedy решения задачи MSkT. Пусть W_i – полный граф с i вершинами. Пусть T_i – k-дерево с i вершинами и $K(T_i)$ – множество всех k-клик в T_i.

АЛГОРИТМ Greedy

Вход: простой взвешенный граф G. Величина $k \geq 2$.

Выход: остовное частичное k-дерево T^* в графе G.

Шаг 0. Определить полный граф G' с множеством вершин $V(G)$ и ребрами, имеющими весами ребер:

$$(\forall i, j \in V(G)) \quad w(i,j) = \begin{cases} w(i,j), \text{ если } [i,j] \in E(G); \\ 0, \text{ если } [i,j] \notin E(G). \end{cases}$$

Шаг 1. Найти ребро $[l,h]^*$ наибольшего веса в графе G'. Построить граф W_2, полагая, что $V(W_2) = \{l,h\}$ и $E(W_2) = [l,h]^*$. Положить $i = 2$.

Шаг 2. (Найти «начальную» клику размера $k+1$ в графе G')

Найти вершину $m^* \in V(G') \setminus V(W_i)$, при которой суммарный вес ребер $\sum_{j \in V(W_i)} w(m^*, j)$, соединяющих такую вершину m^* с вершинами графа W_i максимален. Построить полный граф W_{i+1} путем включения найденной вершины m^* и множества ребер $\{[m^*, j] : j \in V(W_i)\} \subset E(G')$ в граф W_i. Положить $i = i + 1$.

Если $|V(W_i)| = k + 1$, то $T_i = W_i$ и перейти на шаг 3, иначе перейти на шаг 2.

Шаг 3. (Построить k-дерево T_{i+1} размера $i+1$)

Найти вершину $m^* \in V(G') \setminus V(T_i)$ и клику $K^* \in K(T_i)$, при которых

суммарный вес ребер $\sum_{j \in K^*} w(m^*, j)$, соединяющих вершину m^* с вершинами k-клики K^* максимален. Построить k-дерево T_{i+1} путем включения найденной вершины m^* и множества ребер $\{[m^*, j] : j \in K^*\} \subset E(G')$ в k-дерево T_i. Положить $i = i + 1$.

Если $|V(T_i)| = n$, то положить $E(T^*) = E(T_i) \cap E(G)$ и **стоп**, иначе перейти на шаг 3.

Конец алгоритма.

В работе [128] доказано, что трудоемкость эвристики Greedy равна $O(n^3 k)$. Заметим, что эвристика Greedy хотя и не гарантирует нахождения оптимума задачи, но, как показывает практика, строит решение приемлемого качества [128]. Опишем рандомизированую модификацию жадной эвристики Greedy, которая в дальнейшем будет использоваться в разделе 3.2. В данной модификации на шаге $i + 1$ подграф T_{i+1} строится из подграфа T_i, вычисленного на предыдущем шаге i, следующим образом. Для всех вершин $m \in V(G) \setminus V(T_i)$ вычисляется k-клика $K_m \in K(T_i)$ такая, что вес ребер $[m, j] : j \in K_m$ максимален. Подграф T_{i+1} строится из подграфа T_i путем включения в T_i новой вершины $m^* \in V(G) \setminus V(T_i)$ и множества ребер $[m^*, j] : j \in K_{m^*}$, где вершина m^* выбирается с вероятностью

$$\mathbb{P}\{m^* \in V(T_{i+1})\} = \frac{\sum_{j \in K_{m^*}} w(m^*, j)}{\sum_{m \in V(G) \setminus V(T_i)} \sum_{j \in K_m} w(m, j)}.$$

Обозначим Apx$_k$ модификацию приближенного алгоритма Apx, где в качестве «аппроксимирующего» остовного подграфа используется k-дерево. Приведем описание алгоритма Apx$_k$.

Алгоритм Apx$_k$

Вход: простой граф G; множество P; величина k; функции $b(\cdot)$ и $c(\cdot)$.

Выход: отображение π_G вершин графа G в множество P; оценка точности ε.

Шаг 1. С помощью эвристики Greedy найти остовное частичое k-дерево T'

максимального веса в графе G.

Шаг 2. Используя алгоритм [132] построить k-дерево T путем насыщения частичого k-дерева T' ребрами нулевого веса.

Шаг 3. Вычислить оптимальное размещение π_T вершин дерева T в P.

Шаг 4. Положить $\pi_G = \pi_T$ и $\varepsilon = 1 + \Delta(T, \pi_T)/F(T, \pi_T)$.

Стоп.

Конец алгоритма.

Заметим, что поскольку в алгоритме на шаге 2 требуется «насыщать» граф T', найденный эвристикой Greedy, до k-дерева реберами нулевого веса, то для уменьшения времени работы алгоритма Apx_k целесообразно использовать в качестве графа T' не частичное k-дерево, а k-дерево T_i, построенное на последнем шаге эвристики Greedy, которое *уже содержит* необходимые ребра нулевого веса. Подобный подход устраняет необходимоть решения задачи вложения входного частичного k-дерева в k-дерево.

§3.3. Метаэвристика с оценкой точности

В данном разделе предлагается метаэвристика ApxGA сочетающая в себе идеи генетического алгоритма и приближенного алгоритма Apx, предложенного в разделе 3.1. Обозначим $\Pi_i : \Pi_i \subset \Pi$ – популяция особей на итерации i алгоритма ApxGA, где особь $\pi \in \Pi_i$ представляет собой некоторое размещение вершин графа G в множестве позиций P, а Π – множество всех возможных отображений вершин графа G в P. Будем считать, что величина функции $F(G, \pi)$ есть величина приспособленности особи π. Под турнирной селекцией будем понимать такой способ выбора родительских решений, при котором каждое родительское решение выбирается следующим образом: формируется случайное подмножество мощности s (размер турнира) из элементов популяции; среди элементов подмножества выбирается один элемент (родительское решение) с

наименьшим значением целевой функции. В дальнейшем размер турнира s будет полагаться равным 3. Окрестостью $N_1(\pi)$ решения $\pi \in \Pi$ будем называть множество $\{\pi' \in \Pi : \pi \cap \pi' = n - 1\}$ размещений вершин графа G с n вершинами, то есть расстояние Хэмминга между любым элементом окрестности и решением π равно 1. Окрестность Кернигхана-Лина $\mathcal{K}_l(\pi)$ решения π будет определяется следующей последовательностью шагов:

Шаг 0. Положить, что $\pi_0 = \pi$.

Шаг 1. Найти решение π' в окрестности $N_1(\pi_0)$ с минимальным значением целевой функции. Положить $\pi_0 = \pi'$.

Шаг 2. Повторить l раз шаг 1, не рассматривая размещения, уже использованные на предыдущих шагах.

Для построения случайных остовных деревьев в графе G будем использовать следующую рандомизированную модификацию известного алгоритмя Прима. Пусть T_i – дерево с i вершинами. В данной модификации на шаге $i + 1$ подграф T_{i+1} строится из подграфа T_i, вычисленного на предыдущем шаге i, следующим образом. Для всех вершин $m \in V(G) \setminus V(T_i)$ вычисляется вершина $j_m \in V(T_i)$ такая, что вес ребра $[m, j_m] \in E(G)$ максимален. Подграф T_{i+1} строится из подграфа T_i путем включения в T_i новой вершины $m^* \in V(G) \setminus V(T_i)$ и ребра $[m^*, j_{m^*}] \in E(G)$, где вершина m^* выбирается с вероятностью

$$\mathbb{P}\{m^* \in V(T_{i+1})\} = \frac{w(m^*, j_{m^*})}{\sum_{m \in V(G) \setminus V(T_i)} w(m, j_m)}.$$

Алгоритм ApxGA

Вход: простой взвешенный граф G; число особей в популяции r_{max}; вероятность мутации ρ; число итераций I.

Выход: отображение π_G вершин графа G в P; оценка точности ε.

Итерация 0 (формирование начальной популяции).

Положить $r = 1$.

Шаг 1. С помощью рандомизированного алгоритма Прима найти остовное дерево T в графе G.

Шаг 2. Ввычислить оптимальное размещение π_T^* вершин дерева T в P.

Шаг 3. Положить $\pi = \pi_T^*$ и $F'(\pi) = F(T, \pi_T^*)$. Включить π в популяцию Π_0.

Если $r < r_{max}$, то $r = r + 1$ и вернуться на шаг 1, иначе вычислить величину LB нижней оценки оптимума, как $LB = \max_{\pi \in \Pi_0}\{F'(\pi)\}$.

Итерация i: i=1,2,...,I.

Шаг 1 (селекция). Выбрать особи π' и π'' из популяции Π_{i-1} методом турнирной селекции.

Шаг 2 (скрещивание). Построить новую особь π по родительским решениям π' и π'' выполняя шаги 2.0-2.3.

Шаг 2.0. Определить множество позиций размещения $P' = \pi' \cup \pi''$.

Шаг 2.1. С помощью рандомизированного алгоритма Прима найти остовное дерево T в простом графе G.

Шаг 2.2. Вычислить оптимальное решение π_T^* задачи $\langle T, P', F\rangle$.

Шаг 2.3. Положить $\pi = \pi_T^*$.

Шаг 3 (мутация). Модифицировать решение π:

Шаг 3.1. Для каждой вершины $j \in V(G)$ выполнить:

С вероятностью $1 - \rho$ выбрать размещение $\pi(j) \in \pi$. Если $\pi(j)$ выбрано, то размещение $\pi(j)$ вершины j оставить без изменения, иначе заменить в множестве π размещение $\pi(j)$ на $\hat{\pi}(j) \in P$, где $\hat{\pi}(j)$ выбирается с вероятностью

$$\frac{b(j, \hat{\pi}(j)) + \sum_{[j,l]\in E(G)} c([j,l], \hat{\pi}(j), \pi(l))}{\sum_{\pi(j)\in P}(b(j, \pi(j)) + \sum_{[j,l]\in E} c([j,l], \pi(j), \pi(l)))}.$$

Шаг 3.2. Улучшить решение π процедурой наискорейшего локального спуска по окрестности $N_1(\pi)$, а затем по окрестности Кернигхана-Лина $\mathcal{K}_5(\pi)$.

Шаг 4 (обновление популяции). Включить новое решение π в популяцию особей Π_{i-1}. Сформировать новую популяцию $\Pi_i = \Pi_{i-1} \setminus \{\pi'\} : \pi' = \arg\max_{\pi \in \Pi_{i-1}}\{F(G,\pi)\}$. Найти решение π^i с наименьшим весом в Π_i.

Если $i < I$, то $i = i + 1$ и перейти на следующую итерацию i, иначе вычислить наилучшее решение $\pi_G = \pi^{i*} : i^* = \arg\min_i\{F(G,\pi^i)\}$ и оценку точности $\varepsilon = 1 + (F(G,\pi_G) - LB)/LB$.

Стоп.

Конец алгоритма.

Утверждение 3.1. *Вычислительная сложность алгоритма* АрхGA *решения задачи* $\langle G, P, F\rangle$ *равна* $O(n^3(r_{max} + I))$.

Доказательство. На итерации 0 для формирования начальной популяции решений Π_0 требуется $O(n^3 r_{max})$ операций, поскольку трудоемкость алгоритма Арх равна $O(n^3)$, а количество решений вычисленных алгоритмом равно r_{max}. Рассмотрим время выполнения шагов на итерации $i : i = 1, 2, ..., I$. Трудоемкость оператора турнирной селекции равна $O(r_{max})$ и время выполнения операции скрещивания не превосходит $O(n^3)$. На выполнение шага 3.1 для всех вершин $j \in V(G)$ требуется $O(n^2)$ операций. Трудоемкость шага 3.2 равна $O(n^2)$ операций, поскольку трудоемкость процедуры наискорейшего локального спуска по окрестности N_1, $|N_1| = n^2$ равна $O(n^2)$, и трудоемкость поиска по окрестности Кернигхана-Лина $\mathcal{K}_5$ также равна $O(n^2)$. Исходя из этого, трудоемкость одной итерации i, $i = 1, 2, ..., I$ не превосходит $O(n^3)$. Так как количество итераций равно I, тогда вычислительная сложность алгоритма АрхGA равна $O(n^3(r_{max} + I)) = O(n^3 r_{max}) + O(n^3 I)$. **Утверждение 3.1 доказано.**

Приближенный алгоритм Арх используется как на итерации 0 для формирования начальной популяции решений Π_0 и вычисления величины LB нижний границы оптимума, так и на последующих итерациях для нахождения при-

ближенного решения задачи оптимального кроссинговера. Стоит заметить, что имеют место различные модификации вышеприведенного алгоритма, заключающиеся в использовании, как иных операторов селекции, скрещивания, мутации, так и иных способов останова алгоритма.

§3.4. Выводы по третьей главе

В третьей главе представлены приближенные алгоритмы с оценками точности решения ДЗВ. Предложен полиномиальный приближенный алгоритм Apx. Доказана его апостериорная оценка точности решения для случая, когда размещаемый граф является простым. Доказано, что когда для функций b, w, d решаемой задачи выполняется неравенство (3.5), тогда приближенный алгоритм Apx является 2-приближенным. Более того доказано, что когда размещаемый граф является простым циклом, то алгоритм Apx является не только приближенным, но и асимптотически точным. Предложена модификация Apx_k алгоритма Apx, использующая k-дерево в качестве аппроксимирующего подграфа. Построена метаэвристика ApxGA с апостериорной оценкой точности решения ДЗВ, сочетающая в себе идеи предложенного приближенного алгоритма Apx и генетического алгоритма.

Глава 4.

Вычислительные эксперименты

В данной главе представлены результаты вычислительных экспериментов по анализу эффективности предложенных в диссертационной работе как точных, так и приближенных комбинаторных алгоритмов. Эффективность предложенных алгоритмов анализировалась как в сравнении алгоритмов между собой, так и в сравнении с коммерческим программным пакетом IBM ILOG CPLEX Optimization Studio 12.2 и известными в литературе алгоритмами.

§4.1. Исследование эффективности точных алгоритмов

Точные алгоритмы, предложенные в главе 2, были реализованы на ЭВМ в среде MATLAB R2011a. Проведен вычислительный эксперимент по анализу их эффективности. Для оценки эффективности алгоритмов использовался программный пакет IBM ILOG CPLEX Optimization Studio 12.2 (решение модели целочисленного линейного программирования (ЦЛП) дискретной задачи Вебера алгоритмом ветвей и границ). В дальнейшем будем обозначать $\bar{t}_{alg}$ – среднее время работы алгоритма, сек.; $\bar{t}_{ILP}$ – среднее время работы модели ЦЛП, реализованной в IBM ILOG CPLEX, сек. Для тестирования времени работы каждого предложенного алгоритма был сгенерирован класс тестовых задач случайным образом с равномерным распределением, состоящий из серий, каждая из которых включала 20 задач одинаковой размерности. Вычисления проводились на ПК с процессором Intel Core i7 2.6 GHz.

В таблицах 4.1 и 4.2 представлены результаты эксперимента по анализу

времени работы алгоритмов A_C и A_T для случаев ДЗВ, когда размещаемый граф есть k-цепь и k-дерево соответственно.

Таблица 4.1. Результаты тестирования времени работы алгоритма A_C

		Размерность задачи				
Величина k		$n = 5$	$n = 10$	$n = 20$	$n = 40$	$n = 100$
	$\bar{t}_{alg}$	0,0014	0,0041	0,0203	0,2179	3,2854
k=1	$\bar{t}_{ILP}$	0,1421	0,5342	13,1563	—	—
	$\bar{t}_{alg}$	0,0153	0,0224	0,5364	8,6264	354,5622
k=2	$\bar{t}_{ILP}$	0,1625	0,7719	15,4813	—	—
	$\bar{t}_{alg}$	0,0631	0,2526	10,9642	363,2657	—
k=3	$\bar{t}_{ILP}$	0,1924	1,6391	18,5722	—	—
	$\bar{t}_{alg}$	0,1562	2,9752	34,5721	—	—
k=4	$\bar{t}_{ILP}$	0,2121	3,9877	27,3413	—	—
	$\bar{t}_{alg}$	0,2451	35,4758	—	—	—
k=5	$\bar{t}_{ILP}$	0,4499	5,0012	41,4887	—	—

«—» – решение не удалось получить за приемлемое время (10 000 сек.)

Таблица 4.2. Результаты тестирования времени работы алгоритма A_T

		Размерность задачи				
Величина k		$n = 5$	$n = 10$	$n = 20$	$n = 40$	$n = 100$
	$\bar{t}_{alg}$	0,0048	0,0144	0,0758	0,8078	7,8826
k=1	$\bar{t}_{ILP}$	0,1527	0,6408	14,2633	—	—
	$\bar{t}_{alg}$	0,0623	0,1483	1,2792	15,2275	598,2262
k=2	$\bar{t}_{ILP}$	0,1764	0,6960	16,9004	—	—
	$\bar{t}_{alg}$	0,4094	0,9628	23,1001	687,1651	—
k=3	$\bar{t}_{ILP}$	0,1908	0,7408	21,9869	—	—
	$\bar{t}_{alg}$	0,4831	7,7897	82,7618	—	—
k=4	$\bar{t}_{ILP}$	0,2329	1,8105	27,8403	—	—
	$\bar{t}_{alg}$	0,4966	92,3036	—	—	—
k=5	$\bar{t}_{ILP}$	0,3956	3,0078	40,2815	—	—

«—» – решение не удалось получить за приемлемое время (10 000 сек.)

Для задачи Вебера для k-цепи и k-дерева размерности $n = 40$ и выше не удалось получить решение с помощью пакета IBM ILOG CPLEX за приемлемое время для любых значениях параметра k. Заметим, что среднее время решения задачи Вебера такой размерности для 1(2)-цепи алгоритмом A_C не превысило

0,3 (0,9) секунд соответственно, а среднее время решения задачи Вебера данной размерности для 1(2)-дерева алгоритмом A_T не превысило 0,4 (1) секунд соответственно. Отметим, что среднее время решения ДЗВ размерности $n = 100$ для 1-дерева с помощью алгоритма A_T не превысило четырех секунд. Применение точных алгоритмов A_C и A_T при $k \geq 4$ нецелесообразно, так как время работы данных алгоритмов превосходит время работы пакета IBM ILOG CPLEX.

В таблице 4.3 представлены результаты эксперимента по анализу времени работы алгоритма A_S в сравнении с пакетом IBM ILOG CPLEX.

Таблица 4.3. Результаты вычислительного эксперимента

Размерность задачи	$\bar{t}_{alg}$	$\bar{t}_{ILP}$
$n = 5$	0,0167	0,0071
$n = 10$	0,0854	0,6927
$n = 20$	0,9139	15,3793
$n = 40$	9,2132	—
$n = 100$	386,8167	—

«—» – решение не удалось получить за приемлемое время (10 000 сек.)

Для дискретной задачи Вебера для простого цикла размерности $n = 40$ и выше также не удалось получить решение с помощью пакета IBM ILOG CPLEX за приемлемое время, притом, что среднее время решения задачи Вебера такой размерности с помощью предлагаемого алгоритма A_S не превысило десяти секунд. Для задачи Вебера размерности $n = 20$ среднее время работы предлагаемого алгоритма не превысило одной секунды, тогда как среднее время работы пакета IBM ILOG CPLEX составило 16 секунд. Стоит отметить, что среднее время решения ДЗВ размерности $n = 5$ и ниже с помощью точного алгоритма A_S значительно превзошло среднее время работы пакета IBM ILOG CPLEX.

§4.2. Исследование эффективности приближенных алгоритмов

В данном разделе представлены результаты вычислительного эксперимента по анализу эффективности приближенных алгоритмов с оценками точности решения ДЗВ, предложенных в главе 3. Под эффективностью здесь понимается *время работы* алгоритма t, сек., *погрешность* найденного решения δ и его *апостериорная оценка точности* ε. Величина δ погрешности алгоритма рассчитывается по формуле

$$\delta = \frac{F_{alg} - OPT}{OPT},$$

где F_{alg} стоимость приближенного решения, полученное алгоритмом, и OPT – величина оптимума. Входные параметры $w(\cdot)$, $p(\cdot)$ и $d(\cdot)$ тестовых примеров сенерировались случайным образом с равномерным распределением. Все алгоритмы были реализованы в среде MATLAB R2011a. Для вычисления величины OPT точного решения задачи использовался коммерческий программный пакет IBM ILOG CPLEX Optimization Studio 12.2. Вычисления проводились на ПК с процессором Intel Core i7 2.6 GHz.

4.2.1. Анализ эффективности алгоритма Apx

Поскольку величина ε апостериорной оценка точности алгоритма Apx не ограничена константой и зависит от входных данных каждой индивидуальной задачи, то целесообразно провести анализ зависимости величин ε и δ от различных параметров задачи. В настоящей работе в качестве таких параметров, влияющих на значения ε и δ, будем рассматривать следующие. Для задачи $\langle G, P, F\rangle$ – величину n размерности задачи и величину φ «плотности» графа G, определяемую по следующей формуле $\varphi = 2|E(G)|/n(n-1)$, где $E(G)$ – множество ребер в исходном графе G, а $n(n-1)/2$ – максимальное число ребер

в графе с n вершинами. Для задачи $\langle S, P, F\rangle$ – величину n размерности задачи.

В таблицах 4.4 и 4.5 продемонстрированы зависимости средних величин ε и δ алгоритма Apx от величины n размерности задачи $\langle G, P, F\rangle$, а также от величины φ «плотности» графа G. Для каждой пары значений величин n и φ была сгенерирована случайным образом серия из 10-ти тестовых задач.

Таблица 4.4. Зависимость величины оценки ε от параметров n и φ. Задача $\langle G, P, F\rangle$

	$n = 5$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
$\varphi = 0,10$	1,0068	1,0135	1,0386	1,0687	1,0754	**1,0723**
$\varphi = 0,25$	1,1125	1,1235	1,1910	1,2163	1,3016	1,3742
$\varphi = 0,50$	1,3143	1,3855	1,6549	1,9115	2,1780	2,4791
$\varphi = 0,75$	1,4911	1,5512	2,0492	2,3864	2,7135	3,2152
$\varphi = 1,00$	1,6181	1,7897	2,3812	2,9607	3,4073	**4,1261**

Таблица 4.5. Зависимость величины погрешности δ от параметров n и φ. Задача $\langle G, P, F\rangle$

	$n = 5$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
$\varphi = 0,10$	0,0008	0,0015	0,0028	0,0131	0,0153	**0,0173**
$\varphi = 0,25$	0,0219	0,0355	0,0945	0,1649	0,2653	0,3162
$\varphi = 0,50$	0,0574	0,1098	0,2497	0,4431	0,6272	0,7184
$\varphi = 0,75$	0,1355	0,2277	0,5328	0,7433	0,9838	1,2694
$\varphi = 1,00$	0,3081	0,4512	0,8440	1,2723	1,7036	**2,3149**

Легко заметить, что значения величин ε и δ алгоритма Apx возрастают с увеличением значений параметров n и φ, и, соответственно, убывают с уменьшением значения величины φ «плотности» графа G.

Приведем результаты тестирования алгоритма Apx_k, являющегося модификацией приближенного алгоритма Apx, где в качестве «аппроксимирующего» подграфа берется k-дерево. В таблицах 4.6 и 4.7 представлены результаты по анализу зависимости величин оценки ε и погрешности δ алгоритма Apx_k от величины k «аппроксимирующего» k-дерева и величины φ «плотности» графа при фиксированном $n = 30$.

Из таблиц 4.6 и 4.7 видно, что с увеличением величины параметра k, величины ε и δ стремительно уменьшаются, причем, также, чем менее «плотным»

Таблица 4.6. Зависимость величины оценки ε от параметров k и φ при $n = 30$. Задача $\langle G, P, F \rangle$

	$k = 1$	$k = 2$	$k = 3$
$\varphi = 0,10$	1,0889	1,0351	1,0135
$\varphi = 0,25$	1,5314	1,2726	1,0931
$\varphi = 0,50$	2,0552	1,7143	1,2951
$\varphi = 0,75$	2,4952	1,9617	1,4714
$\varphi = 1,00$	3,1996	2,8791	2,2193

Таблица 4.7. Зависимость величины погрешности δ от параметров k и φ при $n = 30$. Задача $\langle G, P, F \rangle$

	$k = 1$	$k = 2$	$k = 3$
$\varphi = 0,10$	0,0139	0,0096	0,0045
$\varphi = 0,25$	0,1714	0,0771	0,0235
$\varphi = 0,50$	0,4352	0,2747	0,0754
$\varphi = 0,75$	0,7452	0,3894	0,1419
$\varphi = 1,00$	1,2891	0,7189	0,2792

является размещаемый граф, тем меньшие значения принимают апостериорная оценка точности ε и погрешность алгоритма δ.

В таблице 4.8 представлена зависимость средних величин ε и δ алгоритма Арх от величины n размерности задачи $\langle S, P, F \rangle$, где S – простой цикл. Для каждого значения n была сгенерирована случайным образом серия из 20-ти тестовых задач. Из таблицы 4.8 видно, что при увеличении размерности зада-

Таблица 4.8. Зависимость величины ε от параметра n. Задача $\langle S, P, F \rangle$

	$n = 10$	$n = 25$	$n = 50$	$n = 75$	$n = 100$	$n = 150$
ε	1,0218	1,0157	1,0071	1,0021	1,0008	1,0002
δ	0,0188	0,0121	0,0053	0,0010	0,0003	0,0000

чи с $n = 10$ до $n = 150$ средняя величина ε апостериорной оценки точности алгоритма Арх уменьшилась с 1,0218 до 1,0002, причем для всех задач размерности $n = 100$ и выше величина ε составила менее 1,0028. Легко заметить что, результаты, полученные в ходе вычислительного эксперимента, согласуются с результатом теоремы 3.4.

В таблицах 4.9 и 4.10 демонстрируются результаты времени работы пред-

ложенного алгоритма Apx в сравнении с пакетом IBM ILOG CPLEX (для простого графа) и известным точным алгоритмом [24] (для простого цикла).

Таблица 4.9. Анализ времени работы алгоритмов Apx и Apx$_k$, сек. Задача $\langle G, P, F\rangle$

	$n = 10$	$n = 25$	$n = 50$	$n = 75$	$n = 100$	$n = 150$
Apx	0,004	0,022	0,218	0,929	3,536	12,504
Apx$_k$ (при k=2)	0,186	2,971	28,518	257,416	633,214	3 815,619
Apx$_k$ (при k=3)	1,462	31,613	691,767	0,929	8 921, 581	—
IBM ILOG CPLEX	3,319	120,046	1 447,714	—	—	—

«—» – решение не удалось получить за приемлемое время (10 000 сек.)

Таблица 4.10. Анализ времени работы алгоритма Apx, сек. Задача $\langle S, P, F\rangle$

	$n = 10$	$n = 25$	$n = 50$	$n = 75$	$n = 100$	$n = 150$
Apx	0,004	0,022	0,218	0,929	3,536	12,504
Точный алгоритм [24]	1,021	3,015	10,071	213,021	367,088	1891,872

«—» – решение не удалось получить за приемлемое время (10 000 сек.)

Как видно из таблиц 4.9 и 4.10, предложенный приближенный алгоритм Apx значительно выигрывает во времени счета у всех представленных точных алгоритмов. Применение алгоритма Apx$_k$ перспективно только для решения задач малой и средней размерности, причем величина параметра k должна быть достаточно мала. Данное решение обусловлено значительным возрастанием времени работы алгоритма, при недостаточном увеличении его точности. Заметим, что алгоритм Apx перспективно исхользовать для решения задачи $\langle S, P, F\rangle$ большой и сверхбольшой размерности, вследствие его быстродействия и асимптотического убывания величины погрешности решения.

4.2.2. Анализ эффективности алгоритма ApxGA

В таблицах 4.11 и 4.12 продемонстрированы зависимости средних величин ε и δ алгоритма ApxGA от величины n размерности задачи $\langle G, P, F\rangle$, а также от величины φ «плотности» графа G. Для каждой пары значений величин n и φ была сгенерирована случайным образом серия из 10-ти тестовых задач. В данном случае полагалось $r_{max} = 30$, $I = 30$ и $\rho = 0,05$.

Таблица 4.11. Зависимость величины ε от параметров n и φ. Задача $\langle G, P, F \rangle$

	$n = 5$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
$\varphi = 0,10$	1,0052	1,0117	1,0256	1,0335	1,0314	**1,0493**
$\varphi = 0,25$	1,0526	1,0682	1,0942	1,1694	1,1967	1,2389
$\varphi = 0,50$	1,2408	1,2644	1,3994	1,4005	1,4016	1,4127
$\varphi = 0,75$	1,3274	1,3328	1,3797	1,4053	1,4108	1,4314
$\varphi = 1,00$	1,3551	1,3739	1,3869	1,3997	1,4225	**1,4353**

Таблица 4.12. Зависимость величины δ от параметров n и φ. Задача $\langle G, P, F \rangle$

	$n = 5$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
$\varphi = 0,10$	0,0000	0,0000	0,0000	0,0007	0,0006	**0,0008**
$\varphi = 0,25$	0,0001	0,0005	0,0016	0,0036	0,0057	0,0081
$\varphi = 0,50$	0,0012	0,0027	0,0045	0,0072	0,0091	0,0121
$\varphi = 0,75$	0,0036	0,0076	0,0089	0,0115	0,0144	0,0187
$\varphi = 1,00$	0,0072	0,0096	0,0113	0,0155	0,0181	**0,0202**

В таблицах 4.13 и 4.14 представлены результаты по анализу эффективности модификации алгоритма ApxGA, где вместо дерева использовалось его обобщение – k-дерево. Ниже представлены зависимости величин ε и δ алгоритма ApxGA от величины от величины k «аппроксимирующего» k-дерева и величины φ «плотности» графа при фиксированном $n = 30$.

Таблица 4.13. Зависимость величины ε от параметров и φ при $n = 30$. Задача $\langle G, P, F \rangle$

	$k = 1$	$k = 2$	$k = 3$
$\varphi = 0,10$	1,0889	1,0215	**1,0089**
$\varphi = 0,25$	1,2418	1,0829	1,0175
$\varphi = 0,50$	1,3952	1,1525	1,0456
$\varphi = 0,75$	1,4071	1,2251	1,1016
$\varphi = 1,00$	1,4096	1,2373	**1,1012**

Таблица 4.14. Зависимость величины δ от параметров k и φ при $n = 30$. Задача $\langle G, P, F \rangle$

	$k = 1$	$k = 2$	$k = 3$
$\varphi = 0,10$	0,0009	0,0002	**0,0000**
$\varphi = 0,25$	0,0038	0,0012	0,0005
$\varphi = 0,50$	0,0071	0,0023	0,0009
$\varphi = 0,75$	0,0121	0,0074	0,0017
$\varphi = 1,00$	0,0151	0,0073	**0,0024**

Было выявлено, что в метаэвристике ApxGA значения величин ε и δ,

также, как в алгоритме Арх, возрастают с увеличением значений n и φ, и убывают с уменьшением значений k и φ. Несмотря на это, значения величин ε и δ в алгоритме АрхGA в среднем в 1,3 и 45 раз меньше, чем в алгоритме Арх соответственно, что говорит о перспективности сочетания приближенного алгоритма Арх с метаэвристиками и алгоритмами локального поиска.

На рисунке 3.1 приведены результаты эксперимента по сравнению эффективности модификаций алгоритма АрхGA с различными вариантов кроссинговера. Полагалось $\varphi = 0,5$. Как видно из рисунка 3.1, решение задачи

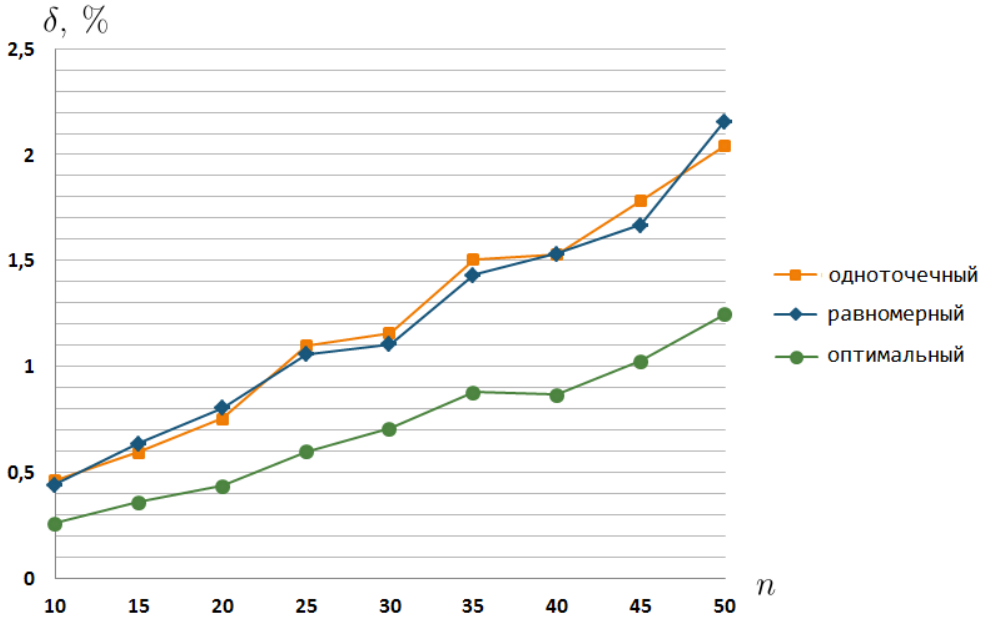

Рис. 4.1. Сравнение эффективности различных вариантов кроссинговера в АрхGA

оптимального кроссинговера на шаге 2 в алгоритме АрхGA приводит к значительно меньшим значениям величин ε и δ, чем использование равномерного и одноточечного кроссинговера.

4.2.3. Сравнение эффективности приближенных алгоритмов

На рисунках 3.2 и 3.3 сравнивается поведение значений величин ε и δ алгоритмов Арх, Арх$_k$ и АрхGA при изменении параметров n и φ задачи $\langle G, P, F \rangle$.

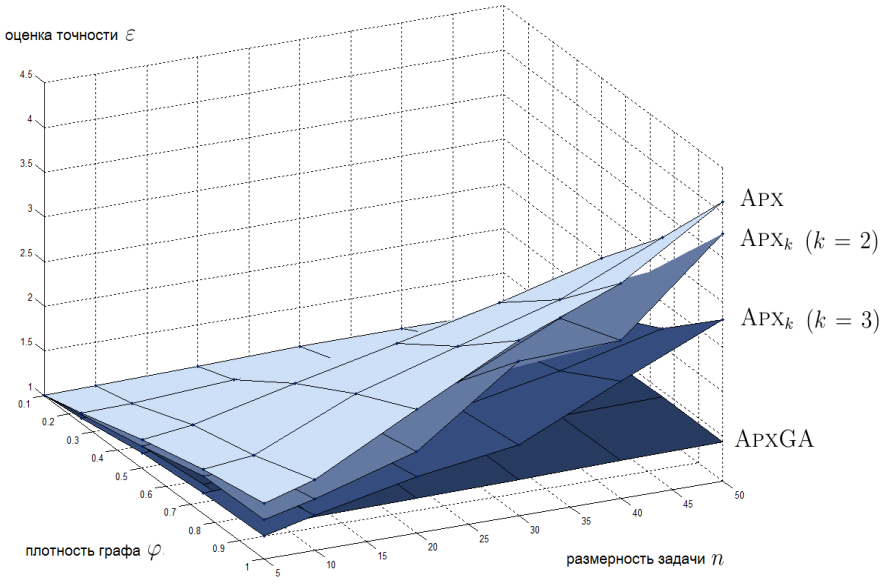

Рис. 4.2. Поведение величины ε алгоритмов Apx, Apx$_k$ и ApxGA при изменении параметров n и φ

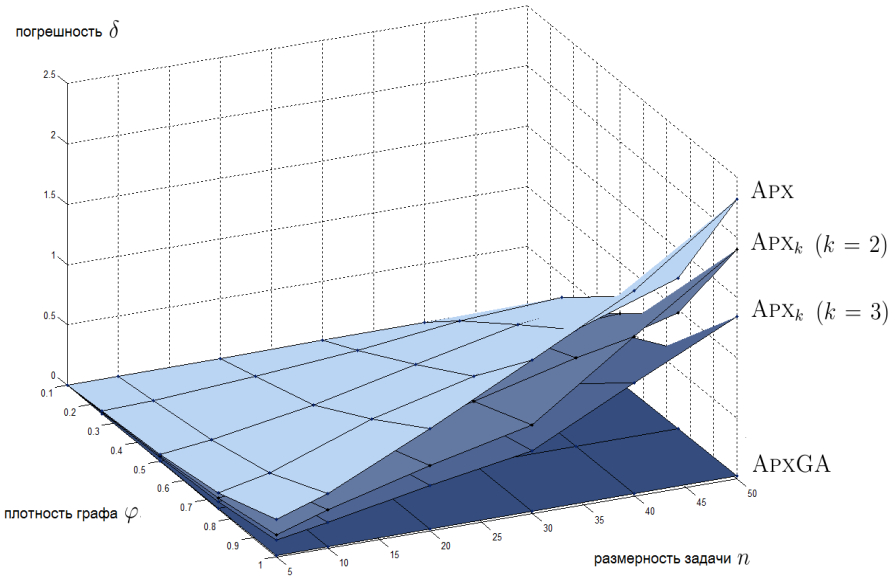

Рис. 4.3. Поведение величины δ алгоритмов Apx, Apx$_k$ и ApxGA при изменении параметров n и φ

В таблице 4.15 представлены результаты сравнения величин нижней границы задачи, полученных алгоритмом Apx, и значений нижней границы задачи, вычисленных с помощью симплекс-метода при решении задачи Вебера с условием ослабления целочисленности переменных. Обозначим λ отношение величины нижней границы задачи, вычисленной алгоритмом Apx, к величине

нижней границы задачи, найденной при решении реаксированной задачи Вебера.

Таблица 4.15. Зависимость величины λ от параметров n и φ. Задача $\langle G, P, F \rangle$

	$n = 5$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$
$\varphi = 0,10$	0,2253	0,2257	0,2656	0,3075	0,3484	**0,3936**
$\varphi = 0,25$	0,3266	0,4125	0,5342	0,6694	0,7497	0,8589
$\varphi = 0,50$	0,4382	0,5752	0,6382	0,7276	0,8235	0,9278
$\varphi = 0,75$	0,6298	0,6925	0,7367	0,8551	0,9151	1,1531
$\varphi = 1,00$	0,9525	1,2953	1,4591	1,6727	1,8451	**2,3277**

Как видно из таблицы 4.15, при значении величины φ «плотности» графа меньше 0,5, значения нижней границы задачи, вычисленные алгоритмом Apx оказались в среднем более близкими к оптимуму, чем величины нижней границы задачи, найденные при решении релаксированной задачи Вебера. Однако, при сравнительно больших значениях величины φ, величина λ в среднем больше единицы, что говорит о том, что значения нижней границы задачи, вычисленные алгоритмом Apx оказались в среднем хуже, чем величины нижней границы задачи, найденные при решении реаксированной задачи Вебера.

В таблице 4.16 демонстрируются результаты эксперимента по сравнению эффективности алгоритмов Apx и ApxGA как между собой, так и в сравнении с известной метаэвристикой поиска с чередующимися окрестностями, предложенной В. Филиповичем и Д. Кратика в работе [97]. Значения величины φ полагалось равным 0,5. Из рисунка 3.4 видно, что предложенная метаэвристи-

Таблица 4.16. Сравнение погрешностей и времени работы алгоритмов

		$n = 10$	$n = 25$	$n = 50$	$n = 75$	$n = 100$
	δ	0,093	0,378	0,739	—	—
Apx	$\bar{t}$	0,004	0,022	0,218	0,929	3,536
	δ	**0,003**	**0,004**	**0,008**	—	—
ApxGA	$\bar{t}$	0,094	0,496	5,721	20,528	89,983
	δ	0,005	0,006	0,022	—	—
Метаэвристика [97]	$\bar{t}$	0,094	0,496	5,721	20,528	89,983
IBM ILOG CPLEX	$\bar{t}$	3,319	120,046	1447,711	—	—

ка АрхGA существенно превосходит известную метаэвристику В. Филиповича и Д. Кратика в точности вычисляемого решения при сравнимом времени счета.

§4.3. Выводы по четвертой главе

В настоящей главе представлены результаты вычислительных экспериментов по анализу эффективности точных и приближенных комбинаторных алгоритмов, предложенных в главах 2 и 3 диссертационной работы.

Проведен эксперимент по анализу времени работы точных алгоритмов, предложенных в главе 2, в сравнении с пакетом IBM ILOG CPLEX. Было выявлено, что применение алгоритмов A_C и A_T для решения ДЗВ перспективно при сравнительно малых значениях величины параметра k, причем, чем меньше величина k и чем больше количество вершин размещаемого графа, тем предложенные алгоритмы более эффективны по сравнению с алгоритмом ветвей и границ, реализованном в пакете IBM ILOG CPLEX. Также справедливо, что чем ближе величина параметра k к числу вершин размещаемого графа, тем предложенные точные алгоритмы A_C и A_T менее эффективны по сравнению с пакетом IBM ILOG CPLEX. Время решения ДЗВ для простого цикла размерности $n = 20$ алгоритмом A_S не превысило одной секунды, тогда как среднее время работы пакета IBM ILOG CPLEX составило 16 секунд, что говорит о перспективности использования алгоритма A_S для решения ДЗВ средней и большой размерности. Стоит заметить, что среднее время решения задачи Вебера размерности $n = 5$ и ниже с помощью предложенного алгоритма A_S значительно превзошло среднее время работы пакета IBM ILOG CPLEX.

Также был проведен эксперимент по анализу приближенных алгоритмов, предложенных в главе 3. Выявлено, что в алгоритмах Арх и АрGA значения величин апостериорной оценки точности ε и относительной погрешности δ возрастают с увеличением значений n и φ. Значения величин ε и δ в алгоритме АрхGA в среднем в 1,3 и 45 раза меньше, чем в алгоритме Арх соответственно.

Применение алгоритма Apx$_k$ перспективно только для решения задач малой и средней размерности, причем величина параметра k должна быть достаточно мала. Данное решение обусловлено значительным возрастанием времени работы алгоритма, при недостаточном увеличении его точности. Проведен эксперимент по сравнению эффективности алгоритмов Apx и ApxGA с известной метаэвристикой поиска с чередующимися окрестностями, предложенной В. Филиповичем и Д. Кратика [97]. Данный эксперимент показал, что предложенная метаэвристика ApxGA существенно превосходит известную метаэвристику в точности вычисляемого решения при сравнимом времени счета. Исходя из результатов эксперимента следует, что алгоритм ApxGA целесообразно использовать как для решения задач средней, так и задач большой размерности, поскольку данный алгоритм находит решение хорошего качества за приемлемое время, причем для каждого вычисленного решения алгоритм позволяет определить насколько далека стоимость найденного решения от оптимума в худшем случае.

Заключение

Настоящая диссертационая работа лежит в области теории оптимизации на графах и посвящена дискретной задаче Вебера. Исследованы свойства дискретной задачи Вебера в графовой постановке. Найдены новые подклассы задачи, которые могут быть разрешены точно либо с гарантированной оценкой погрешности за полиномиальное время. Разработаны эффективные алгоритмы точного и приближенного решения. Предложенные алгоритмы реализованы в виде компьютерных программ, и проверена их эффективность на числовых примерах.

Диссертационное исследование выполнено при поддержке следующих грантов: грант Министерства образования и науки Российской Федерации, соглашение 14.B37.21.0395; грант фонда содействия развитию малых форм предприятий в научно-технической сфере, госконтракт № 11426р/17183.

В заключение перечислим основные полученные результаты диссертационной работы, приведем данные о публикациях и апробациях, и рассмотрим направления дальнейших исследований в данной области.

Основные результаты диссертационной работы

На защиту выносятся следующие новые научные результаты:

1. Сформулирована и доказана теорема о том, что дискретная задача Вебера полиномиально разрешима при фиксированном k на классе k-деревьев. Предложен точный алгоритм решения, основанный на технике динамического программирования по дереву декомпозиции, имеющий оценку времени счета $O(n^{k+3})$ и оценку требуемой оперативной памяти $O(n^{k+3})$, где n – число вершин графа.

2. Сформулирована и доказана теорема о том, что дискретная задача Вебера полиномиально разрешима при фиксированном k на классе k-цепей. Предложен точный алгоритм решения, основанный на технике динамического программирования, имеющий оценку времени счета $O(n^{k+2})$ и оценку требуемой оперативной памяти $O(n^k)$, где n – число вершин графа.

3. Предложен приближенный алгоритм решения дискретной задачи Вебера в случае произвольного размещаемого графа, имеющий трудоемкость $O(n^3)$. Доказана его апостериорная оценка точности и выявлены параметры задачи, влияющие на величину данной оценки. Найдены подклассы задачи, на которых алгоритм является 2-приближенным и асимптотически точным.

4. Для дискретной задачи Вебера с произвольным размещаемым графом предложена метаэвристика, позволяющая вычислять апостериорную оценку погрешности найденного решения. Метаэвристика находит приближенные решения близкие к оптимальным и превосходит в точности решения наилучшую из известных метаэвристик, при сравнимом времени счета.

Публикации по теме диссертации

Статьи в журналах перечня ВАК

1. Шангин Р.Э. Детерминированный алгоритм решения задачи Вебера для n-последовательносвязной цепи // Дискретный анализ и исследование операций. 2013. Т. 20, № 5. С. 84–96.

2. Панюков А.В., Шангин Р.Э. Точный алгоритм решения дискретной задачи Вебера для k-дерева // Дискретный анализ и исследование операций. 2014. Т. 21, № 3. С. 64–75.

3. Шангин Р.Э. Точный и эвристический алгоритмы решения дискретной задачи Вебера для простого цикла // Вестник Новосибирского государственного университета. Серия: матем., механ., информ. 2014. Т. 14, № 2. С. 98-107.

4. Шангин Р.Э. Алгоритм точного решения дискретной задачи Вебера для простого цикла // Прикладная дискр. матем. 2013. № 4, С. 96–102.

5. Шангин Р.Э. Исследование эффективности приближенных алгоритмов решения одного частного случая задачи Вебера // «Экономика, статистика и информатика. Вестник УМО». 2012. № 1. С.163-169.

6. Шангин Р.Э. Точный алгоритм для решения одного частного случая задачи Вебера в дискретной постановке // Прикладная дискретная математика. 2013. № 6. С. 136-137.

Публикации по теме диссертации в других изданиях

7. Шангин Р.Э. Сравнение эффективности приближенных алгоритмов решения одного частного случая задачи размещения // «Статистика. Моделирование. Оптимизация»: Материалы Всероссийской конференции (Челябинск, 26 ноября - 2 декабря, 2011). Челябинск: Изд-во ЮУрГУ. 2011. С. 336-338.

8. Шангин Р.Э. Разработка и анализ алгоритмов для задачи Вебера // «Проблемы оптимизации и экономические приложения»: Материалы V Всероссийской конференции (Омск, 2-6 июля, 2012). Омск: Из-во ОмГУ. 2012. С. 222.

9. Шангин Р.Э. Об одном алгоритме точного решения задачи Вебера для n-последовательносвязной цепи // «Актуальные задачи математического

моделирования и информационных технологий»: Труды международной научно-практической конференции, (Сочи, 25-29 мая, 2013). Сочи: Издательство СГУ. 2013. С. 242-251.

10. Шангин Р.Э. Приближенные алгоритмы решения задачи размещения специального вида // Труды Всероссийской олимпиады по математическим методам и статистике: Сборник трудов участников (Москва, 15-17 ноября, 2011). Москва: Издательство МЭСИ. 2011. С. 120-125.

11. Шангин Р.Э. Об одном алгоритме точного решения задачи Вебера для n-последовательносвязной цепи // «Информационные технологии интеллектуальной поддержки принятия решений»: Труды международной конференции (Уфа, 10-15 апреля, 2013). Уфа: Издательство УГАТУ. 2013. С. 76-81.

12. Шангин Р.Э. Алгоритм точного решения дискретной задачи Вебера для k-дерева // «Высокопроизводительные вычисления - математические модели и алгоритмы»: Труды II Международной конференции (Калининград, 13-17 ноября, 2013). Калининград: Издательство Балтийского федерального университета им. И. Канта. 2013. С. 113-116.

13. Шангин Р.Э. Точный алгоритм решения дискретной задачи Вебера для простого цикла // 45-ая Международная школа-конференция, посвященная 75-летию В.И. Бердышева: Труды Международной школы-конференции (Екатеринбург, 23-26 февраля, 2014). Екатеринбург: Издательство ИММ УрО РАН. 2014. С. 189-192.

14. Шангин Р.Э., Панюков А.В. Алгоритм с оценкой для решения дискретной задачи Вебера // «Математическое программирование и приложения»: Труды XV Всероссийской конференции (Екатеринбург, 2-6 марта, 2015). Екатеринбург: Издательство ИММ УрО РАН. 2015. С. 167-168.

15. Шангин Р.Э., Панюков А.В. Алгоритмы с оценками для решения одной нелинейной задачи о назначениях // Материалы VI Международной конференции «Проблемы оптимизации и экономические приложения» (Омск, 28 июня - 4 июля, 2015). Омск: Из-во ОмГУ. 2015. С. 245.

16. Shangin R.E. Exact algorithm for solving discrete Weber problem for a k-tree // The 15th International Workshop on Computer Science and Information Technologies: Proceedings (Ufa, 4-8 may, 2013). Ufa: USATU. 2013. С. 282-284.

Свидетельство о регистрации программ ЭВМ

17. Шангин Р.Э. Программа нахождения оптимального размещения объектов торгового предприятия на местности // Свидетельство о государственной регистрации программы для ЭВМ № 2014611222 от 28 января 2014 г., правообладатель: ФГБОУ ВПО «ЮУрГУ».

Во всех приведенных выше работах [1-16] научному руководителю А. В. Панюкову принадлежит постановка задачи и общее руководство, Р. Э. Шангину – все полученные результаты.

Направления дальнейших исследований

Теоретические исследования и практические разработки, выполненные в рамках данной диссертационной работы, предполагается продолжить по следующим направлениям.

1. Выделение новых подклассов задачи, для которых возможно построение точных полиномиальных алгоритмов решения.

2. Нахождение новых подклассов задачи, для которых возможно построение приближенных алгоритмов с гарантированными оценками точности и полиномиальных приближенных схем.

3. Построение новых эффективных эвристик и метаэвристик решения дискретной задачи Вебера.

Литература

[1] Быкова В. В. Вычислительные аспекты древовидной ширины графа // Прикладная дискретная математика. 2011. № 3(13). С. 65-79.

[2] Быкова В. В. FTP-алгоритмы на графах ограниченной древовидной ширины // Прикл. дискрет. матем. 2012. № 2(16). С. 65-78.

[3] Васильев И.Л. Метод декомпозиции для задачи о p-медиане на несвязном графе // Дискретн. анализ и исслед. опер. 2007. Т. 14. С. 43–58.

[4] Васильев И.Л., Климентова К.Б. Метод ветвей и отсечений для задачи размещения с предпочтениями клиентов // Дискретный анализ и исследование операций. 2009. Т. 16. № 2. С. 21–41.

[5] Гимади Э. Х., Глебов Н. И., Перепелица В. А. Алгоритмы с оценками для задач дискретной оптимизации // Проблемы кибернетики. М.: Наука. 1975. Т. 31. С. 35-42.

[6] Демиденко В.М. Обобщение условий сильной разрешимости квадратичной задачи о назначениях с матрицами анти-Монжа и Теплица // Доклады НАН Беларуси. 2003. Т. 47. № 2. С. 15-18.

[7] Забудский Г.Г. Задачи оптимального размещения взаимосвязанных объектов: Учебное пособие // Омск: ОмГУ, 2007. 100 С.

[8] Забудский Г. Г., Филимонов Д. В. Решение дискретной минимаксной задачи размещения на сети // Изв. вузов. Матем. 2004. № 5. 33–36.

[9] Забудский Г. Г. Задачи оптимального размещения взаимосвязанных объектов // ОмГУ, 2007. 124 С.

[10] Забудский Г. Г., Лагздин А. Ю. Динамическое программирование для решения квадратичной задачи о назначениях на дереве // АиТ. 2012. № 2. С. 141-–155.

[11] Забудский Г. Г., Лагздин А. Ю. Полиномиальные алгоритмы решения минимаксной квадратичной задачи о назначениях на сетях // Дискр. анализ и исслед. опер. 2011. Т. 18 № 4. С. 49–65.

[12] Забудский Г. Г., Лагздин А. Ю. Полиномиальные алгоритмы решения квадратичной задачи о назначениях на сетях // Журнал вычислительной математики и математической физики. 2010. № 50(11). С. 2052–2059.

[13] Колоколов А. А., Леванова Т. В. Алгоритмы декомпозиции и перебора L-классов для решения некоторых задач размещения // Вестник Омского ун-та. 1996. № 1. С. 21-23.

[14] Панюков А. В. Модели и методы решения задач построения и идентификации геометрического размещения: диссертация на соискание ученой степ. д-ра физ.-мат. наук: 05.13.16. 1999. 260 С.

[15] Панюков А. В., Пельцвергер Б. Ф. Оптимальное размещение дерева в конечном множестве // Журнал вычислительной математики и математической физики. 1988. Т. 28. С. 618-620.

[16] Панюков А. В., Пельцвергер Б. Ф., Шафир А. Ю. Оптимальное размещение точек ветвления транспортной сети на цифровой модели местности // Автом. и Телемех. 1990. № 9. С. 153-162.

[17] Панюков А.В., Шангин Р.Э. Точный алгоритм решения дискретной задачи Вебера для k-дерева // Дискретный анализ и исследование операций. 2014. Т. 21, № 3. С. 64–75.

[18] Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. Москва: Наука. 1984. 387 С.

[19] Перепелица В. А., Гимади Э. Х. К задаче нахождения минимального гамильтонова контура на графе со взвешенными дугами // Дискретный анализ. Новосибирск. 1969. Вып. 15. С. 57–65.

[20] Гимади Э. Х., Перепелица В. А. Асимптотически точный подход к решению задачи коммивояжера // Управляемые системы. Сб. науч. тр. Новосибирск: Ин-т математики СО АН СССР. 1974. Вып. 12. С. 35–45.

[21] Сергеев С. И. Квадратичная задача назначения I. Новые нижние границы в схеме парного назначения // АиТ. 1999. № 8. С. 127-147.

[22] Сергеев С. И. Квадратичная задача назначения II. Улучшенный алгоритм Гилмора–Лоулера // АиТ. 1999. № 9. С. 137-–143.

[23] Трубин В. А. Эффективный алгоритм для задачи Вебера с прямоугольной метрикой // Кибернетика. 1978. № 6. С. 67-70.

[24] Шангин Р.Э. Алгоритм точного решения дискретной задачи Вебера для простого цикла. Прикладная дискретная математика. 2013. № 4. С. 96–102.

[25] Шангин Р.Э. Детерминированный алгоритм решения задачи Вебера для n-последовательносвязной цепи // Дискретный анализ и исследование операций. 2013. Т. 20, № 5. С. 84–96.

[26] Ahuja R. K., Orlin J. B., Tivari A. A greedy genetic algorithm for the quadratic assignment problem // Working paper 3826-95, Sloan School of Management, MIT. 1995.

[27] Andrews G. Foundations of multithread, parallel, and distributed programming // Addison Wesley. 2000.

[28] Armstrong R. D, Jin Z. Solving linear bottleneck assignment problems via strong spanning trees // Operations Research Letters. 1992. Vol. 12. P. 179–180.

[29] Arkin E., Hassin R., Sviridenko M. Approximating the maximum quadratic assignment problem // Inf. Process. Lett. 2001. Vol. 77. P. 13–16.

[30] Arnborg S. Efficient Algorithms for Combinatorial Problems with Bounded Decomposability - A Survey // BIT Numerical Mathematics. 1985. Vol. 25(1). P. 1-23.

[31] Arnborg S., Proskurowski A. Linear time algorithms for NP-hard problems restricted to partial k-trees // Discr. math. 1987. Vol.6(3). P. 278–291.

[32] Arnborg S., Corneil D.G., Proskurowski A. Complexity of Finding Embeddings in a k-Tree // SIAM. Journal on Algebraic and Discrete Methods. 1989. Vol. 8(2). P. 277–284.

[33] Avella P., Sassano A., Vasilev I. Computational study of large-scale p-median problems // Mathematical Programming. 2007. Vol. 109. P. 89–114.

[34] Balas E., Saltzman M. An algorithm for the three-index assignment problem // Operations Research. 1991. Vol. 39. P. 150-161.

[35] Bandelt H. J., Crama Y, Spieksma F. Approximation algorithms for multidimensIonal assignment problems with decomposable costs // Discrete Applied Mathematics. 1994. Vol. 49. P. 25-50.

[36] Beck H., Candia A. An heuristic for the minimum spanning 2-tree problem // Computer Science. Vol. 47. P. 97-109. 1993.

[37] Beck H., Candia A., Bravo H. Optimal design of invulnerable networks // Research Report. Vol. 15. P. 107–113. 1993.

[38] Beck H., Candia A. Heuristics for minimum spanning k-trees // Investigation Operativa. 2000. Vol. 9. P. 104-116.

[39] Beltran H., Skorin-Kapov D. On minimum cost isolated failure immune network // Telecommunications System Modeling and Analysis. 1993. Vol. 12. P. 444-453.

[40] Bern M. W. Networks Design Problems: Steiner Trees and Spanning k-Trees // Ph. D. Thesis. University of Berkeley. 1987.

[41] Bern M. W., Lawler E. L., Wong A. L. Linear-time computation of optimal subgraphs of decomposable graphs // Journal of Algorithms. 1987. Vol. 8(2). P. 216–235.

[42] Bodlaender H. L. A partial k-arboretum of graphs with bounded treewidth // Theoretical Computer Science. 1998. Vol. 20. P. 1–45.

[43] Boese K., Kahng A., Muddu S. A new adaptive multi-start technique for combinatorial global optimizations // Oper. Res. Lett. 1994. Vol. 16. P. 101-114.

[44] Bokhari S. H. A shortest tree algorithm for optimal assignments across space and time in a distributed processor system // IEEE Trans. Software Engrg. 1981. Vol. 7. P. 441-456.

[45] Boyera Vol. Baza D., Elkihel M. Solving knapsack problems on GPU // Computers & Operations Research. 2012. Vol. 39. P. 42–47.

[46] Brimberg J., Drezner Z., Mladenovic N., Salhi S. A new local search for continuous location problems // European Journal of Operational Research. 2014. Vol. 232. P. 256-265.

[47] Burkard R. E. Cela E. Linear Assignment Problems and Extensions // Handbook of Combinatorial Optimization. 1999. Vol. 1. P. 75-149.

[48] Burkard R., Cela E. Quadratic and three-dimensional assignments: An annotated bibliography // Combinatorial Optimization. 1998. Vol. 4. P. 373-392.

[49] Burkard R. E., Cela E., Pardalos P.M., Pitsoulis L.S. The Quadratic Assignment Problem // Handbook of Combinatorial Optimization. 1998. Vol. 2. P. 241-238.

[50] Burkard R., Rendl F. A thermodynamically motivated simulation procedure for combinatorial optimization problems // European Journal of Operational Research. 1984. Vol. 17. P. 169-174.

[51] Cai L., Maffray F. On the spanning k-tree problem // University of Toronto press. 1992.

[52] Cai L. On spanning 2-trees in a graph // University of Hong Kong. 1996.

[53] Calamai P., Conn A. A stable algorithm for solving the multifacility location problem involving Euclidean distances // SIAM J. Sci. Statist. Comput. 1996. Vol. 1. P. 512-525.

[54] Candia A., Bravo H. A Simulated Annealing Approach for Minimum Cost Isolated Failure Immune Networks // Essays and Surveys in Metaheuristics. 2002. Vol. 15. P. 169-183.

[55] Casti J., Richardson M., Larson R. Dynamic programming and parallel computers // Journal of Optimization. Theory and Applications. 1973. Vol. 12. P. 423-438.

[56] Chakrapani J. Skorin-Kapov J. Massively parallel tabu search for the quadratic assignment problem // Annals of Operations Research. 1993. Vol. 41. P. 327–342.

[57] Cheng K. Y. Note on minimizing the bandwidth of sparse, symmetric matrices // Computing. 1973. Vol. 11. P. 27-30.

[58] Chung F. R. K., Seymour P. D. Graphs with small bandwidth and cutwidth // Discrete Mathematics. 1989. Vol.75. P. 113-119.

[59] Connolly D. T. An improved annealing scheme for the QAP // European Journal of Operational Research. Vol. 46. 1990. P. 93–100.

[60] Conway D., Venkataramanan, M. Genetic search and the dynamic facility layout problem // Computers & Operations Research. 1994. Vol. 21. P. 955-960.

[61] Collins R. J. Bandwidth reduction by automatic renumbering // International Journal for Numerical Methods in Engineering. 2005. Vol. 6. P. 345-356.

[62] Colorni A., Maniezzo Vol. The ant system applied to the quadratic assignment problem // IEEE Transactions on Knowledge and Data Engineering. 1999. Vol. 11. P. 769-778.

[63] Crama Y. and Spieksma F. Approximation algorithms for three-dimensional assignment problems with triangle inequalities // European Journal of Operational Research. 1992. Vol. 60. P. 273-279.

[64] Cristofides N. Graph Theory. An Algorithm Approach // New York: Academic Press. 1975.

[65] Kruskal J. B. On the shortest spanning subtree of a graph and the traveling salesman problem // Proceedings of the American Mathematical Society. 1956. Vol. 7. P. 48-–50.

[66] Domschke W. Schedule synchronization for public transit networks // OR Spektrum. 1989. № 11. P. 17-24.

[67] Domschke W., Forst P., Voss S. Tabu search techniques for the quadratic semi-assignment problem // New Directions for Operations Research in Manufacturing. 1992. V. 3. P. 389-405.

[68] Durmaz E., Aras N., Kuban Altinel I. Discrete approximation heuristics for the capacitated continuous location–allocation problem with probabilistic customer locations // Computers & Operations Research. 2009. Vol. 36. P. 2139-2148.

[69] Farley A. Networks immune to isolated failures // Networks. 1981. Vol. 11. P. 255–268.

[70] Fernandez de la Vega W., Lamari M. The taskallocation problem with constant communication // Discrete Applied Mathematics. 2003. Vol. 131. P. 169-–177.

[71] Fleurent C., Ferland J. Genetic hybrids for the quadratic assignment problem // DIMACS Series in Discrete Mathematics and Theoretical Computer Science 1994. Vol. 16. P. 173–187.

[72] Fliege J. Coincidence conditions in multifacility location problems with positive and negative weights // European Journal of Operational Research. 1998. Vol. 104. P. 310-320.

[73] Francis R. L., Cabot A. Vol. Properties of a multifacility location problem involving Euclidean distances // Naval Res. Log. Quart. 1972. Vol. 19. P. 335-353.

[74] Francis R. L., McGinnis L. F., White J. A. Futility Layout and Location: An Analytical Approach // Prentice Hall, Englewood Cliffs, NJ. 1991.

[75] R. Fulkerson, Glicksberg I., Gross O. A production line assignment problem // Technical Report RM–1102. The Rand Corporation. 1953.

[76] Gallo G., Tomasin E. M., Sorato A. M. Lower bounds for the quadratic semi-assignment problem. Rutgers University, New Brunswick, NJ. 1986

[77] Ghashghai E., Rardin R. Using genetic algorithms to find good k-tree subgraphs // Evolutionary Optimization. 2002. Vol. 48. P. 399-413.

[78] Golumbic M. Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York. 1980.

[79] Granot D., Skorin-Kapov D. On some optimization problems on k-trees and partial k-trees. Discrete Applied Mathematics. 1994. Vol. 48. 129-145.

[80] Grotschel M., Monma C. and Stoer M. Facets for polyhedra arising in the design of communication networks with low-connectity constraints // SIAM J. Optim. 1992. Vol. 2. P. 474-504.

[81] Gross O. The bottleneck assignment problem // Technical Report P–1630, The Rand Corporation, 1959.

[82] Hansen, P. and Jaumard, B. Cluster Analysis and Mathematical Programming // Mathematical Programming. 1997. Vol.79. P. 191-215.

[83] Hansen P., Mladenovic N., Perez-Brito D. Variable neighbourhood decomposition search // J. Heuristics. 2001. Vol. 7. P. 335–350.

[84] Hansen P., Mladenovic N., Brimberg J., Moreno Perez J. Handbook of Metaheuristics // International Series in Operations Research Management Science. 2010. Vol. 146. P. 61-86.

[85] Mladenovic N., Brimberg J., Hansen P., Moreno-Perez J. A. The p-median problem: a survey of metaheuristic approaches // European J. Oper. Res. 2007. Vol. 3. P. 927–939.

[86] Hansen P., Mladenovic N., Brimberg J., Urosevic D. Primal-Dual Variable Neighborhood Search for the Simple Plant-Location Problem // INFORMS Journal on Computing. 2007. Vol. 19. P. 552-564.

[87] Hansen P., Mladenovic N. J-Means: a new local search heuristic for minimum sum of squares clustering // Pattern Recognition. 2001. Vol. 34. P. 405–413.

[88] Hansen P., Mladenovic N. Variable neighborhood search for the p-median // Location Science. 1997. Vol. 5. P. 207–226.

[89] Hansen H., Mladenovic N., Taillard E. Heuristic solution of the multisource Weber problem as a p-median problem // Operations Research Letters. 1998. Vol. 22. P. 55-62.

[90] Ibarra L. The clique-separator graph for chordal graphs // Discrete Applied Mathematics. 2009. Vol. 157(8). P. 1737-1749.

[91] Jarre F. On the convergence of the method of analytic centers when applied to convex quadratic programs // Math. Programming. 1991. Vol. 49. P. 341-358.

[92] Kaplan H. Shamir R. Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques // SIAM J. Comput. 1996. Vol. 25, № 3, P. 540-561.

[93] Kariv O., Hakimi L., An algorithmic approach to network location problems. II: The p-medians // SIAM J. Appl. Math. 1979. Vol. 37. P. 539–560.

[94] Karzanov A.Vol. One more well-solved case of the multifacility location problem // Discrete Optimization. 2004. Vol. 1. P. 51-66.

[95] Karzanov A.Vol. Hard cases of the multifacility location problem // Discrete Applied Mathematics, 2004. Vol. 143. P. 368-373.

[96] Kinnersley N. G. The vertex separation number of a graph equals its path-width // Information Processing Letters. 1992. Vol. 42. P. 345–350.

[97] Kratica J., Filipovich V at al. Solving the Task Assignment Problem with a variable neighborhood search // Serdica J. Computing. 2010. Vol. 4. P. 435–446.

[98] Koopmans T. C., Beckmann M. Assignment problems and the location of economica activities // Econometrica. 1957. Vol. 25. P. 53–76.

[99] Krarup, J. and Pruzan, P. The simple plant location problem: survey and synthesis // European J. Oper. Res. 1983. Vol. 12. P. 36-81.

[100] Kuenne R., Soland R. Exact and approximate solutions to the multisource Weber problem // Math. Programming. 2001. Vol.3. P. 193-209.

[101] Kuhn H. W. The Hungarian Method for the assignment problem // Naval Research Logistics Quarterly. 1955. Vol. 2. P. 83—97.

[102] Lawler E. L., Lenstra J. K., Rinnoy Kan A. H. G., Shmoys D. B. The traveling salesman problem: A guided tour of combinatorial optimization // John Wiley & Sons, Chichester. 1985.

[103] Levin Y., Ben-Israel A. A heuristic method for large-scale multi-facility location problems // Computers & Operations Research. 2004. Vol. 31. P. 257-272.

[104] Li Y., Pardalos P. M., and Resende M. G. C. A greedy randomized adaptive search procedure for the quadratic assignment problem // DIMACS Series in Discrete Mathematics and Theoretical Computer Science. 1994. Vol. 16. P. 237–261.

[105] Livesley R. K. The Analysis of Large Structural Systems // The Computer Journal. 1960. Vol. 1. P. 34-39.

[106] Luis M., Salhi S., Nagy G. A guided reactive GRASP for the capacitated multi-source Weber problem // Computers & Operations Research. 2011. Vol. 38. P. 1014-1024.

[107] Magos D. Tabu search for the planar three-index assignment problem // Journal of Global Optimization. 1996. Vol. 8. P. 35-48.

[108] Magos D., Miliotis P. An algorithm for the planar three-index assignment problem // European Journal of Operational Research. 1994. Vol. 77. P. 141-153.

[109] Malucelli F. Quadratic Assignment Problems: Solution Methods and Applications. Ph.D. Thesis. University of Pisa. 1993.

[110] Malucelli F. A polynomially solvable class of the quadratic semi-assignment problems // European J. Oper. Res. 1996. Vol. 91. P. 619-622.

[111] Malucelli F., Pretolani D. Quadratic semi-assignment problem on structured graphs // Ric. Oper. 1994. Vol. 69. P. 57-78.

[112] Malucelli F., Pretolani D. Lower bounds for the quadratic semi-assignment problem // European Journal of Operational Research. 1995. Vol. 83. P. 365-375.

[113] Malucelli F., Pretolani D. Lower bounds for the quadratic semi-assignment problem // European J. Oper. Res. 1995. Vol. 83. P. 365-375.

[114] McKee T. A. On the Chordality of a Graph // Journal of Graph Theory. 1993. Vol. 17. P. 221-232.

[115] Megiddo N., Supowit K.J. On the complexity of some common geometric location problems // SIAM J. Comput. 1984. Vol. 13. P. 182-196.

[116] Panyukov A. Vol., Pelzwerger B. Vol. Polynomial Algorithms to Finite Veber Problem for a Tree Network // Journal of Computational and Applied Mathematics. 1991. Vol. 35. P. 291-296.

[117] Pardalos P. M., Pitsoulis L. Nonlinear Assignment Problems: Algorithms and Applications // Springer US. 2000.

[118] Pochet Y., Wolsey L.A. Production Planning by Mixed Integer Programming // NewYork:Springer. 2006.

[119] Prim R. Shortest connection networks and some generalizations // Bell Systems Techn. Journal. 1957. Vol. 36. P. 1389-1401.

[120] Reese J. Solution methods for the p-median problem: An annotated bibliography // Networks. 2006. Vol. 48 P. 125-142.

[121] Resende M. G. C., Werneck R. F. A grasp with path-relinking for the p-median problem // Technical Report TD-5E53XL, AT&T Labs. 2002.

[122] Robertson N., Seymour P. D. Graph minors. II. Algorithmic aspects of treewidth // J. Algorithms. 1986. Vol. 7. P. 309-322.

[123] Robertson N., Seymour P. D. Graph Minors. XX. Wagner's conjecture // Journal of Combinatorial Theory, Series B. 2004. Vol. 92. P. 325–357.

[124] Rose D. J. On Simple Characterizations of k-trees // Discrete Mathematics. 1973. Vol. 7. P. 317-322.

[125] Rose D. J. Triangulated Graphs and the Elimination Process // J. of Math. Analysis and Appl. 1970. Vol. 32. P. 597-609.

[126] Sahni S., Gonzalez T. P-complete Approximation Problems // Journal of the ACM (JACM). 1976. Vol. 23(3). P. 555-565.

[127] Skorin-Kapov J. Tabu search applied to the quadratic assignment problem // ORSA Journal on Computing. 1990. Vol. 2. P. 33–45.

[128] R.E. Shangin, P.M. Pardalos. Heuristics for Minimum Spanning K-tree Problem // Procedia Computer Science. 2014. Vol. 31. P. 1074–1083.

[129] Smith K. Solving the generalized quadratic assignment problem using a self-organizing process // Proceedings of the IEEE International Conference on Neural Networks (ICNN). 1995. Vol. 4. P. 1876-1879.

[130] Stone H. S. Multiprocessor scheduling with the aid of networkIow algorithms // IEEE Trans. Software Engrg. SE-3(1). 1977. P. 85-–93.

[131] Takahashi, A., Ueno S., Kajitani Y. Minimal acyclic forbidden minors for the family of graphs with bounded path-width // Discrete Mathematics. 1994. Vol. 127. P. 293–304.

[132] Tansel B. C., Francis R. L., Lowe T. J. Location on Networks: A Survey. Part II: Exploiting Tree Network Structure // Management Science. 1983. Vol. 29. P. 498-511.

[133] Taillard E. Robust tabu search for the quadratic assignment problem // Parallel Computing. 1991. Vol. 17. P. 443–455.

[134] Tate D. M. and Smith A. E. A genetic approach to the quadratic assignment problem // Computers and Operations Research. 1995. Vol. 22. P. 73–83.

[135] Urban T. Solution procedures for the dynamic facility layout problem // Annals of Operations Research. 1998. Vol. 76. P. 323-342.

[136] Voss S. Network design formulations in schedule synchronization // Computer-Aided Transit Scheduling. 1992. Vol. 386. P. 137-152.

[137] Voss S. Tabu search: applications and prospects. // Network Optimization Problems. 1993. V. 1. P. 333-353.

[138] Wald J., Colbourn C. Steiner trees, partial 2-trees and minimum IFI networks // Networks. 1983. Vol. 13. P. 159-167.

[139] Weber A. Uber den Standort der Industrien, Teil 1: Reine Theorie des Standortes // Tubingen: J.C.B. Mohr. 1909.

[140] Xue G. L., Rosen J. B., Pardalos P. M. A polynomial time dual algorithm for the Euclidean multifacility location problem // Operations Research Letters. 1996. Vol. 18. P. 201-204.

[141] Zainuddin Z., Salhi S. A perturbation-based heuristic for the capacitated multisource Weber problem // European Journal of Operational Research. 2007. Vol. 179. P. 1194-1207.

Приложение. Основные обозначения

№ п/п	Обозначение	Значение	Раздел
1	G	простой взвешенный граф	1.3
2	$V(G)$	множество вершин графа G	1.3
3	$E(G)$	множество ребер графа G	1.3
4	n	число вершин в графе	1.3
5	$w(i,j)$	вес ребра $[i,j]$	1.3
6	P	множество позиций размещения	1.3
7	$d(t,u)$	расстояние между позициями t и u	1.3
8	π	отображение $V(G)$ в P	1.3
9	c	функция стоимости размещения ребра	1.3
10	b	функция стоимости размещения вершины	1.3
11	$F(G,\pi)$	стоимость размещения π графа G	1.3
12	π^*	оптимальное размещение графа	1.3
13	$\mathcal{T}$	дерево декомпозиции	2.1.1
14	ε	оценка точности алгоритма	3.1
15	δ	погрешность алгоритма	4.2
16	φ	величина плотности графа	4.2