ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

ПАНИН Артем Александрович

СЛОЖНОСТЬ И АЛГОРИТМЫ РЕШЕНИЯ ДВУХУРОВНЕВЫХ ЗАДАЧ РАЗМЕЩЕНИЯ И ЦЕНООБРАЗОВАНИЯ

01.01.09 — Дискретная математика и математическая кибернетика

Диссертация на соискание ученой степени
кандидата физико–математических наук

Научный руководитель
к.ф.–м.н. А.В. Плясунов

Новосибирск — 2015

Оглавление

Введение

Актуальность темы. Задачи размещения производства и ценообразования имеют широкий круг приложений в производственной, промышленной, сырьедобывающей и многих других сферах деятельности. Задачи размещения предприятий возникают при планировании и реконструкции производства, проектировании сетей обслуживания, в стандартизации, кластерном анализе и других областях. Задачи ценообразования стали неотъемлемой частью современной рыночной экономической модели. Исследования в области задач размещения ведутся в Институте математики им. С.Л. Соболева СО РАН с конца 60-х годов прошлого столетия. Актуальность этих исследований обусловлена их важными практическими приложениями. Об этом свидетельствует большое число работ, посвященных задачам размещения (как в конкурентной среде, так и при отсутствии противников). Среди них в первую очередь стоит отметить работы Береснева В.Л., Гимади Э.Х., Дементьева В.Т., Гермейера Ю.Б., Шамардина Ю.В., Колоколова А.А., Антипина А.С., Хамисова О.В., Васильева И.Л., Забудского Г.Г., Левановой Т.В. и др. Касательно задач ценообразования следует выделить работы Дементьева В.Т., Шамардина Ю.В., Григорьева А.Ю., Свириденко М.И. и др. Из зарубежных авторов касательно задач размещения и ценообразования нужно

обратиться к исследованиям Aboolian R., Berman O., Krass D., Dasci A., Laporte G., Serra D., ReVelle C., Eiselt H.A., Drezner T., Marianov V., van Loon J. и др. В настоящее время данная область активно развивается. Устанавливается и исследуется вычислительная сложность задач в рамках классической теории. Развиваются точные и приближенные методы решения.

Цель диссертации. Установление вычислительной сложности исследуемых задач размещения производства и ценообразования. Исходя из этого, разработка и исследование точных и приближенных методов решения.

Объектом исследования диссертации являются задачи размещения производства и ценообразования. Предмет исследования – сложность данных задач и алгоритмы их решения.

Методы исследования. В диссертации использованы современные методы исследования операций, включающие в себя построение математических моделей, классическую теорию сложности, в том числе, полиномиальную и аппроксимационную иерархии классов сложности, а также методологию экспериментальных исследований с применением вычислительной техники и коммерческих пакетов прикладных программ для решения задач частично-целочисленного, целочисленного, булевого двухуровневого программирования.

Научная новизна. Оригинальность и научная новизна полученных результатов состоит в следующем:

1) Предложены новые модели ценообразования, размещения и ценообразования, конкурентного размещения и ценообразования, государст-

венно-частного партнерства.

2) Исследована сложность задач ценообразования с различными ценовыми стратегиями. Установлено, что задачи дискриминационного и равномерного ценообразования полиномиально разрешимы, а задача фабричного ценообразования NP-трудна в сильном смысле и принадлежит классу $Log\text{-}APX$.

3) Для задачи фабричного ценообразования разработаны эффективные гибридные алгоритмы, основанные на методе декомпозиции и метаэвристиках.

4) Исследована сложность задач размещения и ценообразования с различными стратегиями ценообразования и типами размещения предприятий. Установлено, что все они являются NP-трудными в сильном смысле и принадлежат классу $Poly\text{-}APX$, причем задачи с одним из типов размещения (когда за размещение взимается плата) являются $Poly\text{-}APX$-полными относительно AP-сводимости, а это значит, что для них не существует приближенного эффективного алгоритма с относительной погрешностью "лучше" полиномиальной при условии $P \neq NP$.

5) Для задачи размещения и фабричного ценообразования, когда открывается заданное число предприятий, разработаны эффективные гибридные алгоритмы на основе генетического локального поиска и спуска с чередующимися окрестностями.

6) Исследована сложность задачи конкурентного размещения и ценообразования. Установлено, что она является Σ_2^P-трудной и лежит в классе $Poly\text{-}APX_2^P$. Для нее разработаны эффективные алгоритмы, основывающиеся на идеях альтернирующей эвристики и локального поис-

ка.

7) Исследована сложность задачи государственно-частного партнерства. Установлено, что она является NPO-трудной. Для нее разработан эффективный гибридный алгоритм, основанный на методе локального поиска.

Личный вклад. Основные научные результаты получены автором лично. Вклад соискателя заключается в исследовании сложности поставленных задач, разработке и анализе точных и приближенных алгоритмов, их реализации и проведении вычислительного эксперимента. Представление изложенных в диссертации результатов, полученных в совместных исследованиях, с соавторами согласовано.

Практическая и теоретическая ценность. Работа носит теоретический и экспериментальный характер. Исследована сложность задач размещения и ценообразования. Для их решения разработаны точные и приближенные методы. Данные методы реализованы в виде программ. Они показали свою эффективность и могут применяться при решении практических задач, а также использоваться в университетских курсах «Исследование операций» и «Теория принятия решений».

Апробация работы. Все разделы диссертации прошли апробацию на следующих конференциях в России и за рубежом:

- V Международная азиатская молодежная школа по оптимизации больших систем, Иссык-Куль, 2009;
- Международная конференция «Дискретная оптимизация и исследование операций», Алтай, 2010, и Новосибирск, 2013;
- Международная конференция «Optimization and applications»

(OPTIMA2011), Петровац, Черногория, 2011 г;

- Всероссийская конференция «Проблемы оптимизации и экономические приложения», Омск, 2012;
- Международная конференция «ISMP2012», Берлин, Германия, 2012;
- Международная конференция «EURO2013», Рим, Италия, 2013;
- XVI Байкальская международная школа-семинар «Методы оптимизации и их приложения», о. Ольхон, 2014;
- XV-я Всероссийская конференция «Математическое программирование и приложения», Екатеринбург, 2015;
- Научные семинары Института математики им. С.Л. Соболева СО РАН.

Публикации. По теме диссертации автором опубликовано 15 работ, в том числе 6 статей в журналах из списка ВАК.

Объем и структура диссертации. Диссертация состоит из введения, пяти глав, заключения и списка литературы (85 наименования). Объем диссертации – 140 страница.

СОДЕРЖАНИЕ РАБОТЫ

Во введении формулируются цель и задачи исследования, обосновывается актуальность выбранной темы и указываются основные методы решения поставленной задачи. Отмечена новизна полученных результатов и их практическая и теоретическая ценность. Приводятся сведения об апробации работы и публикациях. Кратко излагается содержание работы.

В **первой главе** формулируются новые модели ценообразования, размещения и ценообразования, конкурентного размещения и ценообразования, государственно-частного партнерства, исследуемые в диссертационной работе. Описываются базовые классы сложности, относительно которых определяется вычислительная сложность рассматриваемых постановок.

В работе рассматриваются двухуровневые модели. Область двухуровневых задач оптимизации является неотъемлемой частью теории экстремальных задач. Она корнями уходит в работы Штакельберга [78] и широко используется для изучения разного рода иерархических постановок, которые можно содержательно описать с помощью следующей игры. В ней участвуют два игрока, принимающие решения последовательно – сначала первый игрок (лидер), а затем второй (конкурент). При этом поведение каждого из игроков определяется некоторой оптимизационной задачей. Одна из них описывает выбор решения первым игроком (верхний уровень), а вторая моделирует реакцию второго игрока на данное решение (нижний уровень). При этом может возникнуть проблема для лидера, когда задача нижнего уровня имеет несколько решений. Здесь мы рассматриваем только кооперативные постановки, когда конкурент среди всех своих оптимальных решений выбирает "лучшее" с точки зрения лидера.

Сформулируем содержательную постановку исследуемых задач ценообразования в виде игры Штакельберга. На верхнем уровне производитель выбирает цены на каждом из своих предприятий, выпускающих однородную продукцию. На нижнем уровне каждый из потребителей выби-

рает одно из предприятий, на котором транспортные затраты и затраты на приобретение продукции в сумме минимальны. Покупка совершается в том случае, когда это позволяет бюджет потребителя. Здесь предполагается единичный спрос. Требуется определить такие цены на каждом предприятии, при которых доход производителя от продажи продукции максимален.

Предлагаемая стратегия ценообразования называется фабричной (mill pricing). Помимо нее в диссертационной работе рассматриваются следующие две стратегии. Равномерное ценообразование (uniform pricing), т.е. на всех пунктах обслуживания устанавливается одна и та же цена. Дискриминационное ценообразование (discriminator pricing) – стратегия ценообразования, при которой могут быть ущемлены интересы каких-то групп покупателей, т.к. на каждом пункте обслуживания могут устанавливаться разные цены для разных покупателей. Соответственно, в зависимости от выбора стратегии ценообразования рассмотрим задачи фабричного, равномерного и дискриминационного ценообразования.

В выше описанных моделях ценообразования предприятия (магазины) уже размещены. Производителю остается только назначить цены на производимую продукцию. Если в данной игре Штакельберга положить, что сперва производителю необходимо разместить предприятия, то получим модель размещения производства и ценообразования. В диссертации рассматриваются два типа размещения предприятий: **I** тип – когда за размещение предприятия взимается определенная плата, **II** тип (иногда называется медианным) – когда необходимо разместить определенное количество предприятий, при этом плата за размещение не взимается. В

зависимости от выбора одной из трех стратегий ценообразования и типа размещения получим шесть задач размещения и ценообразования, которые являются обобщением выше описанных задач ценообразования.

Помимо моделей размещения и ценообразования в работе рассматривается модель конкурентного размещения и ценообразования, описываемая следующей игрой Штакельберга. Задано конечное множество пунктов размещения предприятий и конечное множество рынков. Предполагается, что предприятия производят однородный продукт. Первым ходом лидер размещает свои предприятия, затем свой выбор делает конкурент. Каждому игроку необходимо открыть определенное число предприятий, т.е. используется тип размещения II. Для каждого рынка и предприятия известна общая себестоимость производства и доставки товара потребителям рынка. После того как выбраны предприятия процесс ценообразования на каждом рынке реализуется на основе модели ценовой конкуренции Бертрана [71]. В этой модели игроки конкурируют между собой изменяя цены на продукцию стремясь к себестоимости производимой продукции. В результате ценовой конкуренции рынки будут поделены между лидером и конкурентом. Предприятие лидера захватывает рынок, если себестоимость поставляемого им продукта минимальна среди всех открытых предприятий. Оставшиеся рынки захватываются предприятиями конкурента. Монополист на каждом своём рынке, в отличии от модели Бертрана, устанавливает оптимальную монопольную цену и получает прибыль равную произведению величины спроса на разность монопольной цены и себестоимости. Прибыль игрока складывается из прибыли с каждого из монополизированных им рынков. Цель игры

лидера – выбрать такое множество пунктов размещения при заданном бюджетном ограничении, которое позволяет монополизировать рынки, доставляющие максимальную суммарную прибыль.

В дополнение рассмотрена модель государственно-частного партнерства, в которой государство и инвестор вступают в игру, конкурируя между собой, по освоению минерально-сырьевой базы. При этом они реализуют три группы проектов: инвестиционные (производственные), инфраструктурные и экологические. Реализацию проектов можно интерпретировать как размещение предприятий. Сперва лидер (государство) реализует инфраструктурные и экологические проекты, затем конкурент (инвестор) запускает инвестиционные и, если нужно, экологические. При этом оба игрока имеют бюджетные ограничения на реализацию соответствующих проектов. Цель игры – реализовать такие инфраструктурные и экологические проекты, что после ответа инвестора государство получит наибольшую прибыль.

Во **второй главе** определяется сложностной статус задач ценообразования. Показано, что задачи равномерного и дискриминационного ценообразования полиномиально разрешимы. Касательно задачи фабричного ценообразования имеют место следующие результаты:

Теорема 5 *Задача фабричного ценообразования является NP-трудной в сильном смысле.*

Теорема 4 *Задача фабричного ценообразования полиномиально разрешима при выполнении любого из следующих условий:*

1) Число потребителей фиксировано;

2) Число предприятий фиксировано.

Теорема 6 *Задача фабричного ценообразования принадлежит классу Log-APX.*

Теорема 7 *При условии $P \neq NP$ для задачи фабричного ценообразования не существует полиномиального приближенного алгоритма с абсолютной погрешностью, ограниченной константой.*

Первые две теоремы характеризуют сложность нахождения оптимального решения. Из теоремы 5 следует, что исследуемая задача ценообразования с фабричной стратегией относится к классу труднорешаемых задач, но (как следствие теоремы 4) при выполнении определенных условий она становится эффективно разрешимой. Теорема 6 говорит нам о том, что для задачи фабричного ценообразования существует полиномиальный алгоритм с логарифмической относительной погрешностью. Но при этом, скорее всего, не удастся построить эффективный алгоритм с константной абсолютной погрешностью (теорема 7).

В конце предлагается гибридный алгоритм решения задачи фабричного ценообразования, на основе метода декомпозиции и метаэвристик, в частности, генетического локального поиска, в том числе его различные модификации для уменьшения трудоемкости.

В **третьей главе** исследуется сложность задач размещения производства и ценообразования с различными типами размещения и стратегиями ценообразования. Получены следующие результаты:

Теорема 8 *Исследуемые задачи размещения и ценообразования NP-трудны в сильном смысле.*

Теорема 9 *Задачи размещения и ценообразования, в которых за размещение взимается плата, полиномиально разрешимы в случае фикси-*

рованного числа возможных мест открытия предприятий. А задачи размещения и ценообразования, когда требуется открыть известное число предприятий, полиномиально разрешимы в случае фиксированного числа открываемых предприятий.

Теорема 10 *Задачи размещения и ценообразования, в которых за размещение взимается плата, являются Poly-APX-полными относительно AP-сводимости.*

Из последней теоремы следует, что при несовпадении классов P и NP не существует эффективного приближенного алгоритма с относительной погрешностью лучше полиномиальной для этих задач.

Для задачи размещения и фабричного ценообразования, в которой требуется открыть известное число предприятий, разработаны гибридные алгоритмы, основанные на идеях генетического локального поиска, поиска с чередующимися окрестностями и локального поиска. Вычислительный эксперимент показал их конкурентоспособность в сравнении с известными методами решения.

В **четвертой главе** приводятся результаты по сложности задачи конкурентного размещения производства и ценообразования:

Теорема 11 *Параметрическая задача конкурента является NP-трудной в сильном смысле.*

Теорема 12 *Задача конкурентного размещения и ценообразования является Σ_2^P-трудной.*

Теорема 13 *Задача конкурентного размещения и ценообразования принадлежит классу $Poly\text{-}APX_2^P$.*

Для решения данной задачи разработаны алгоритмы, основывающи-

еся на идеях альтернирующей эвристики и локального поиска, основной недостаток которых связан с их быстрым зацикливанием.

В **пятой главе** исследуется задача государственно-частного партнерства. Получены следующие результаты:

Теорема 14 *Задача государственно-частного партнерства является NPO-трудной.*

Теорема 15 *Параметрическая задача инвестора является NPO-полной.*

Для решения задачи государственно-частного партнерства разработаны приближенные алгоритмы, основанные на идеях локального поиска.

В **заключении** приводятся основные результаты диссертации.

Публикации автора по теме диссертации:

- Панин А., Плясунов А. Задача ценообразования. Ч. 1. Точные и приближённые алгоритмы решения // Дискрет. анализ и исслед. операций. —2012. —т. 19. № 5. —С. 83-100.

- Панин А., Плясунов А. Задача ценообразования. Ч. 2. Вычислительная сложность // Дискрет. анализ и исслед. операций. —2012. —т. 19. № 6. —С. 56-71.

- Панин А., Плясунов А. О сложности двухуровневых задач размещения и ценообразования// Дискретный анализ и исследование операций. —2014. —том 21, № 5. —С. 54–66.

- Кочетов Ю., Панин А., Плясунов А. Сравнение метаэвристик для решения двухуровневой задачи размещения предприятий и фабричного ценообразования // Дискретный анализ и исследование опера-

ций. —2015. —том 22, № 3. —С. 36–54.

- Панин А., Пащенко М., Плясунов А. Двухуровневые модели конкурентного размещения производства и ценообразования // АиТ. —2014. —№ 4. —С. 153–169.

- Лавлинский С., Панин А., Плясунов А. Двухуровневая модель планирования государственно-частного партнерства // АиТ. —2015. —№ 11.

- Панин А.А. Генетический алгоритм для одной задачи ценообразования // Труды ИВМиМГ СО РАН серия Информатика. —Новосибирск: URSS, 2009. —Вып. 9. —С. 190–196.

- Панин А.А. Верхние оценки для одной задачи ценообразования // Российская конференция «Дискретная оптимизация и исследование операций»: Материалы конференции. —Новосибирск: Изд-во Ин-та математики, —2010. —С. 117.

- Panin A.A., Plyasunov A.V. Computational complexity and decomposition algorithms for the mill pricing problem // Abstract of II International conference "Optimization and applications"(OPTIMA-2011). —Petrovac, —2011, —P. 46–49.

- Панин А.А., Плясунов А.В. О сложности задачи размещения и ценообразования // «Проблемы оптимизации и экономические приложения»: материалы V Всероссийской конференции. — Омск, Изд-во Ом. гос. ун-та, —2012. —С. 158.

- Panin A. On approximability some location and pricing problems //

Abstract of 21st International Symposium on Mathematical Programming (ISMP2012). —Berlin, —2012, —P. 101.

- Панин А.А., Пащенко М.Г., Плясунов А.В. Новая модель конкурентного размещения и ценообразования // Российская конференция «Дискретная оптимизация и исследование операций»: Материалы конференции. —Новосибирск: Изд-во Ин-та математики, —2013. —С. 117.

- Panin A., Plyasunov A. Approximate algorithms for the bilevel facility location and pricing problem // 26th European Conference on Operational Research (EURO2013), —Rome, —2013, —P. 284.

- Панин А.А. Метаэвристики для одной задачи размещения и ценообразования // Тезисы XVI Байкальской международной школы-семинара «Методы оптимизации и их приложения», —Иркутск, —2014, —С. 80.

- Панин А.А., Плясунов А.В. О некоторых моделях размещения предприятий и ценообразования // XV-я Всероссийская конференция «Математическое программирование и приложения»: Тезисы. —Екатеринбург, —2015, —С. 153–154.

Глава 1

Постановки задач размещения производства и ценообразования

Двухуровневые задачи оптимизации стали неотъемлемой частью теории экстремальных задач. Начиная с работы Штакельберга [78], двухуровневое программирование широко используется для изучения разного рода иерархических постановок, которые можно содержательно описать с помощью следующей игры. В ней участвуют два игрока, принимающие решения последовательно – сначала первый игрок, а затем второй. При этом поведение каждого из игроков определяется некоторой оптимизационной задачей. Одна из них описывает выбор решения первым игроком, а вторая моделирует реакцию второго игрока на данное решение. Иерархические задачи такого вида возникают в разных областях экономики, техники, военного дела и других сферах деятельности [1, 8, 18, 40, 70].

Большое разнообразие как существующих, так и возникающих новых двухуровневых постановок настоятельно диктует необходимость развивать теорию двухуровневого программирования, в том числе, разрабатывать новые методы решения таких задач. Обзор результатов можно

найти в ряде монографий, среди которых наиболее известна [41]. Большая часть этих результатов связана с непрерывными постановками и лишь сравнительно небольшая доля посвящена смешано-целочисленным задачам. За последние годы разработаны новые точные методы решения двухуровневых задач со смешанными переменными, основанные на методах отсечений [26, 42]. Много интересных результатов получено в области разработки быстрых приближённых алгоритмов, использующих метаэвристики [5, 6, 12, 21, 22, 31, 36, 43, 51]. Появились исследования, в которых разрабатываются гибридные алгоритмы, в той или иной форме совмещающие возможности точных и приближённых методов решения двухуровневых задач [22, 24]. Интересные результаты получены в области изучения вычислительной сложности таких задач [38, 70].

Одна из основных проблем, которая возникает в процессе исследования любой оптимизационной задачи – понять, насколько эффективно она может быть решена численно. В идеале, это понимание достигается с помощью определения верхней границы (на основе разработки эффективного алгоритма решения), а также определением нижней границы. Если верхняя и нижняя границы совпадают, то проблема решена. Таким образом, теоретически оптимальный алгоритм найден. В то время как в исследовании верхних границ достигнуты значительные успехи, то о нижних границах сложности известно относительно мало [4, 25, 27]. В связи с отсутствием эффективных методов для получения точных нижних границ, был развит очень полезный и мощный инструмент, связанный с понятиями сводимости, полноты и сложности классов [4, 25, 27].

Напомним обозначения, используемые в теории сложности при опи-

сании полиномиальной иерархии классов сложности [4, 25, 27]. Первые два базовых класса задач распознавания P и NP определяются с помощью детерминированных и недетерминированных машин Тьюринга [25]. Класс P образован задачами, которые распознаются за полиномиальное время на детерминированных машинах Тьюринга. Соответственно, класс NP определяется как класс задач, которые распознаются за полиномиальное время на недетерминированных машинах Тьюринга. Третий базовый класс *co-*NP состоит из задач распознавания, чьи дополнения лежат в классе NP. Данные классы образуют первый уровень полиномиальной иерархии, и их обозначают как Δ_1^P, Σ_1^P и Π_1^P соответственно. Второй уровень полиномиальной иерархии определяется с помощью детерминированных и недетерминированных оракульных машин Тьюринга [25]. Задача распознавания L принадлежит классу Δ_2^P, если существует детерминированная оракульная машина Тьюринга, которая распознает за полиномиальное время задачу L, используя в качестве оракула некоторый язык из класса NP. Класс Δ_2^P часто обозначают как P^{NP}. Аналогично определяется класс Σ_2^P, в котором задачи распознаются недетерминированными машинами Тьюринга с NP-оракулом, и его дополнение – класс Π_2^P. По определению $P = \Delta_1^P \subseteq \Sigma_1^P \cap \Pi_1^P = NP \cap$ *co-*NP и $NP \cup$ *co-*$NP = \Sigma_1^P \cup \Pi_1^P \subseteq \Delta_2^P \subseteq \Sigma_2^P \cap \Pi_2^P$. Известно, что если $NP \neq$ *co-*NP, то приведённые включения являются строгими [25]. На рисунке 1 дана диаграмма, где приводятся все включения между классами иерархии уровней 1 и 2.

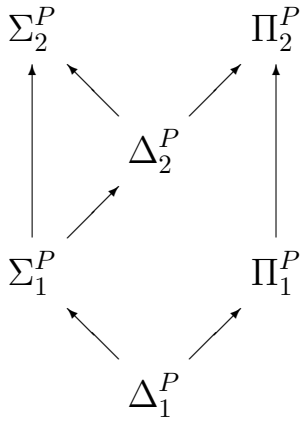

Рисунок 1: Полиномиальная иерархия (1–2 уровень).

Полиномиальная иерархия относится к сложности задач распознавания, с помощью которых оценивается сложность нахождения оптимального решения задач оптимизаций. Имеет смысл рассмотреть вопрос нахождения "неплохого" допустимого решения. Обычно в этом случае рассматривают сложность задачи с точки зрения построения эффективного алгоритма нахождения приближенного решения с гарантированной оценкой точности, т.е. положение оптимизационной задачи в иерархии аппроксимационных классов [27]:

$$PO \subseteq FPTAS \subseteq PTAS \subseteq APX \subseteq \textit{Log-APX} \subseteq$$

$$\subseteq \textit{Poly-APX} \subseteq \textit{Exp-APX} \subseteq NPO.$$

Каждый из этих классов описывает определённое качество аппроксимации, которым обладают образующие его оптимизационные задачи. Данная иерархия используется для описания свойств задач из класса NPO. Содержательно его можно описать как класс оптимизационных задач, у которых стандартная задача распознавания принадлежит классу NP. Здесь под стандартной задачей распознавания, соответствующей

оптимизационной задаче, подразумевается проблема, в которой требуется определить для некоторого α существует или нет допустимое решение со значением целевой функции большим или равным α [4]. Класс PO образован задачами, для каждой из которых существует точный полиномиальный алгоритм решения. Класс $FPTAS$ состоит из задач, для которых существуют вполне полиномиальные приближенные схемы решения, а класс $PTAS$ образован задачами, для которых существуют полиномиальные приближенные схемы решения. Классы APX, $Log\text{-}APX$, $Poly\text{-}APX$, $Exp\text{-}APX$ состоят из задач, для которых существуют полиномиальные приближенные алгоритмы решения, соответственно, с константной, с логарифмической, с полиномиальной и с экспоненциальной оценками точности погрешности. В последних трех случаях значения указанных функций зависят от длины записи исходных данных задачи. Формальные определения можно найти в [27, 28]. Также известно, что при условии $P \neq NP$ указанные выше включения между классами являются строгими [27, 28, 33].

1.1 Задачи ценообразования

В моделях исследования операций, имеющих экономическое содержание, ценовые или затратные параметры, как правило, предполагаются заданными. Основное внимание уделяется поиску наилучших в том или ином смысле объемов производства, планов поставок, размещений пунктов производства, номенклатур изделий и т.п. Процессы ценообразования в условиях рыночных отношений выдвигают новые задачи, целью которых является нахождение цен на продукцию, наилучших для ее про-

изводителя [7, 30].

Сформулируем содержательную постановку исследуемых задач ценообразования, которые отличаются друг от друга только выбором стратегии ценообразования, в виде игры Штакельберга [41]. На верхнем уровне производитель выбирает цены на каждом из своих предприятий, выпускающих однородную продукцию. На нижнем уровне каждый из потребителей выбирает одно из предприятий, на котором транспортные затраты и затраты на приобретение продукции в сумме минимальны. Покупка совершается в том случае, когда это позволяет бюджет потребителя. Требуется определить такие цены на каждом предприятии, при которых доход производителя максимален.

Предлагаемая стратегия ценообразования называется фабричной (mill pricing) [57]. Помимо нее обычно рассматриваются следующие две стратегии. Равномерное ценообразование (uniform pricing) [57], т.е. на всех пунктах обслуживания устанавливается одна и та же цена. Дискриминационное ценообразование (discriminator pricing) - стратегия ценообразования, при которой могут быть ущемлены интересы каких-то групп покупателей, т.к. на каждом пункте обслуживания могут устанавливаться разные цены для разных покупателей [57]. Соответственно, в зависимости от выбора стратегии ценообразования рассмотрим задачи фабричного, равномерного и дискриминационного ценообразования.

Задачи ценообразования – довольно обширная область. Касательно этих проблем следует выделить работы Дементьева В.Т., Шамардина Ю.В., Григорьева А.Ю., Свириденко М.И., Aboolian R., Berman O., Krass D., Dasci A., Laporte G., Serra D., ReVelle C., van Loon J. [7, 20, 30, 34, 55,

76] и др.

Для формализации задачи, описания математической модели введем следующие обозначения:

$I = \{1, ..., n\}$ – множество пунктов производства (предприятий);

$J = \{1, ..., m\}$ – множество потребителей (клиентов);

$b_j \geq 0, j \in J$ – бюджет (ценовой порог) j-го потребителя;

$c_{ij} \geq 0, i \in I, j \in J$ – матрица транспортных затрат потребителей;

$p_i \geq 0, i \in I$ – цена товара на i-м предприятии;

$$x_{ij} = \begin{cases} 1, \text{если } i\text{-е предприятие обслуживает } j\text{-го потребителя,} \\ 0, \text{в противном случае.} \end{cases}$$

Используя данные обозначения запишем задачу фабричного ценообразования в виде следующей модели двухуровневого квадратичного программирования:

$$\sum_{i \in I} \sum_{j \in J} p_i x_{ij} \to \max_{p,x}$$
$$p_i \geq 0; i \in I;$$

где вектор x – оптимальное решение задачи нижнего уровня:

$$\sum_{j \in J} \sum_{i \in I} (b_j - c_{ij} - p_i) x_{ij} \to \max_{x}$$
$$\sum_{i \in I} x_{ij} \leq 1; j \in J;$$
$$x_{ij} \in \{0, 1\}; i \in I, j \in J.$$

Целевая функция задачи определяет доход производителя. Целевая функция нижнего уровня выражает величину сэкономленного потребителями бюджета, а ограничения гарантируют, что каждый потребитель

обслуживается не более чем одним предприятием производителя. Также из этих ограничений и определения целевой функции следует, что покупка совершается только в том случае, когда это позволяет бюджет потребителя.

Если величины c_{ij} принимают только два значения: либо 0, либо $+\infty$, и в двухуровневой задаче требуется, что каждый потребитель должен быть удовлетворен, то получим наиболее близкую к исследуемой постановку, рассмотренную в [7].

Введем дополнительные обозначения для описания постановок задач дискриминационного и равномерного ценообразования Пусть $p_{ij} \geq 0$ – цена товара на i-м предприятии для j-го потребителя, а $p \geq 0$ – цена товара (на всех предприятиях и для всех потребителей одинакова). Тогда назовем задачей дискриминационного ценообразования задачу фабричного ценообразования, в которой вместо переменной p_i используется переменная p_{ij}, а задачей равномерного ценообразования будем называть задачу фабричного ценообразования, в которой вместо переменной p_i используется переменная p.

Под оптимальным решением двухуровневой задачи можно понимать любое её допустимое решение, на котором достигается максимум целевой функции задачи. В целом это удовлетворительное определение. Однако, возникают проблемы, когда задача нижнего уровня имеет несколько оптимальных решений, которые с точки зрения потребителей равнозначны. В этом случае производитель, выбрав наилучшим образом размещение пунктов обслуживания и приемлемые с точки зрения потребителей цены, может недосчитаться прибыли. Это связано с тем, что в данной ситуации

потребители могут выбрать такой вариант поведения, который является оптимальным как решение внутренней задачи, т.е. каждый из них максимально экономит свой бюджет, но при этом хотя бы один из потребителей оказывается в пункте обслуживания с меньшей ценой, чем ожидал производитель. Таким образом данный выбор не оказывается оптимальным с точки зрения производителя. Чтобы избежать подобной ситуации, предположим, что в случае нескольких оптимальных решений в задаче нижнего уровня каждый потребитель выбирает тот пункт обслуживания (из числа доступных), который ближе к нему. Содержательно, это означает, что потребители выбирают такое оптимальное решение, которое сохраняет прибыль производителя. Другими словами рассматриваются кооперативные постановки задач. Отметим, что в постановке рассмотренной в [7] такой проблемы не возникает. В силу специфичного выбора транспортных затрат не возникает разницы между кооперативной и не кооперативной постановками задачи.

Предположения о том, что потребители подыгрывают производителю, оказывается также достаточным, чтобы записать двухуровневую задачу фабричного ценообразования в виде следующей задачи квадратичного программирования со смешанными переменными MP (Mill pricing problem):

$$\sum_{i\in I}\sum_{j\in J} p_i x_{ij} \to \max_{p,x} \tag{1.1}$$

при ограничениях:

$$\sum_{i\in I}(b_j - c_{ij} - p_i)x_{ij} \geq 0; j \in J \tag{1.2}$$

$$\sum_{i \in I}(c_{ij} + p_i)x_{ij} \leq c_{kj} + p_k; k \in I, j \in J \tag{1.3}$$

$$\sum_{i \in I} x_{ij} \leq 1; j \in J; \tag{1.4}$$

$$p_i \geq 0, x_{ij} \in \{0, 1\}; i \in I, j \in J. \tag{1.5}$$

Как и ранее, целевая функция (1.1) определяет доход производителя. Ограничения (1.2) гарантируют, что потребитель не выйдет за рамки своего бюджета. Выполнение ограничений (1.3) приводит к тому, что транспортные затраты потребителя и его затраты на приобретение продукции в сумме минимальны. Ограничения (1.4) означают, что каждый потребитель может быть обслужен не более чем в одном предприятии. По аналогии заменой ценовых переменных определяются задачи дискриминационного ценообразования DP (Discriminator pricing problem) и равномерного ценообразования UP (Uniform pricing problem).

1.2 Задачи размещения и ценообразования

Постановки, связанные с размещением производства и/или ценообразования, образуют широкий спектр математических моделей, методов и приложений в исследовании операций [30, 44, 45, 50, 55, 83]. Процессы размещения и ценообразования обычно исследуются отдельно и независимо друг от друга [30, 50, 55, 83]. Основная причина этого связана с тем, что они относятся к разным горизонтам планирования. Процессы размещения являются долговременными, а процессы ценообразования относятся к краткосрочному планированию. Как следствие, в большинстве случаев сначала выбирается размещение, а затем цены. Однако еще в [57] отмечалось, что разделение размещения и ценообразования неприемлемо

из-за того, что размещение предприятий (магазинов) должно проводиться с учётом имеющегося спроса, который, так или иначе, зависит от цен. С другой стороны, выбор наилучших цен зависит от того, где расположены пункты производства. Таким образом, разделение размещения и ценообразования при моделировании приводит к невозможности получить наилучшие варианты размещения и цен. Более того, даже в ситуации, когда нет необходимости знать точные цены на производимый продукт, а достаточно знать лишь диапазон цен в выбранной нише рынка, позволяющих противостоять конкурентам, разделение размещения и ценообразования также нецелесообразно [20]. Поэтому современные подходы к выбору эффективного механизма взаимодействия процессов размещения производства и ценообразования основываются на их совместном анализе в рамках одной модели [19, 20, 34, 76]. Однако, для оценки качества принимаемых решений, необходимо уметь адекватно оценивать и реакцию рынка, в частности, потребителей на предлагаемый вариант размещения и ценообразования. И с этой целью удобно моделировать весь процесс в виде задачи двухуровневого программирования [9, 41].

Рассмотрим следующую игру Штакельберга. В игре участвуют два типа игроков: лидер (производитель) и последователи (потребители). Первым делает ход производитель. Он открывает (размещает) предприятия по производству однородной продукции и назначает цену на производимый товар. Затем каждый потребитель выбирает такое открытое предприятие, на котором его суммарные затраты на покупку и транспортировку товара минимальны, и совершает покупку в том случае, если эти затраты не превышают его бюджет. Цель игры: открыть такие

предприятия и установить такие цены, при которых доход производителя (сумма цен на товары, купленные потребителями) за вычетом затрат на открытие предприятий максимален.

Далее мы будем рассматривать лишь кооперативный вариант выше описанной игры, как и в случае задач ценообразования. Т.е. предположим, что если затраты какого-нибудь из потребителей минимальны в нескольких предприятиях, то он выберет то предприятие, которое ближе к нему, т.е. с точки зрения производителя будет выбрано предприятие с наибольшей ценой товара. Ко всему прочему, ограничимся рассмотрением только фабричной, равномерной и дискриминационной стратегий ценообразования, описанных ранее. Также будем использовать два типа размещения предприятий: **I** и **II**. В типе I за размещение предприятия с производителя взимается определенная плата, а в типе II потребителю необходимо открыть фиксированное число предприятий, при этом плата за их размещение не взимается.

Пусть теперь множество I обозначает множество возможных мест открытия предприятий. Введем обозначения в дополнение к тем, что использовались в задачах ценообразования:

$f_i \geq 0$ – стоимость открытия предприятия в месте $i \in I$ (i-го предприятия);

$r \in N$ – число открываемых предприятий;

$$y_i = \begin{cases} 1, \text{если } i\text{-е предприятие открыто,} \\ 0, \text{в противном случае.} \end{cases}$$

Эти обозначения будут использоваться в постановках шести иссле-

дуемых задач размещения и ценообразования: LDP (location and discriminator pricing problem), LMP (location and mill pricing problem) и LUP (location and uniform pricing problem), которые соответствуют всем трем выше описанным стратегиям ценообразования и типу размещения I, а также $MLDP$ (median location and discriminator pricing problem), $MLMP$ (median location and mill pricing problem) и $MLUP$ (median location and uniform pricing problem), если используется тип II.

Используя данные обозначения запишем игру Штакельберга для случая дискриминационного ценообразования и типа размещения I в виде следующей задачи двухуровневого квадратичного программирования (LDP):

$$\sum_{i\in I}\sum_{j\in J} p_{ij}x_{ij} - \sum_{i\in I} f_i y_i \to \max_{p,x,y}$$
$$p_{ij} \geq 0; y_i \in \{0,1\}; i \in I; j \in J;$$

где вектор x – оптимальное решение задачи нижнего уровня:

$$\sum_{i\in I}\sum_{j\in J}(b_j - c_{ij} - p_{ij})x_{ij} \to \max_{x}$$
$$\sum_{i\in I} x_{ij} \leq 1; j \in J;$$
$$x_{ij} \leq y_i; i \in I, j \in J;$$
$$x_{ij} \in \{0,1\}; i \in I, j \in J.$$

Целевая функция задачи определяет доход производителя. Целевая функция нижнего уровня выражает величину сэкономленного потребителями бюджета, а ограничения гарантируют, что каждый потребитель

обслуживается не более чем одним предприятием производителя, которое должно быть открыто. Также из этих ограничений и определения целевой функции следует, что покупка совершается в том случае, когда это позволяет бюджет потребителя.

На основе задачи LDP опишем задачи LMP и LUP. Назовем задачей размещения и фабричного ценообразования LMP задачу LDP, в которой вместо переменной p_{ij} используется переменная p_i, а задачей размещения и равномерного ценообразования LUP будем называть задачу LDP, в которой вместо переменной p_{ij} используется переменная p.

Также, используя введенные обозначения, запишем игру Штаккельберга для случая дискриминационного ценообразования и типа размещения II в виде следующей задачи двухуровневого квадратичного программирования ($MLDP$):

$$\max_{y,p,x} \sum_{i\in I}\sum_{j\in J} p_{ij}x_{ij}$$

$$\sum_{i\in I} y_i = r;$$

$$p_{ij} \geq 0, y_i \in \{0,1\}; i \in I, j \in J,$$

где вектор x – оптимальное решение задачи нижнего уровня:

$$\max_{x} \sum_{i\in I}\sum_{j\in J} (b_j - c_{ij} - p_{ij})x_{ij}$$

$$\sum_{i\in I} x_{ij} \leq 1; j \in J;$$

$$x_{ij} \leq y_i; i \in I, j \in J;$$

$$x_{ij} \in \{0,1\}; i \in I, j \in J.$$

Целевая функция двухуровневой задачи определяет доход компании, а ограничение гарантирует, что будет открыто ровно r предприятий. Целевая функция нижнего уровня описывает стратегию каждого потребителя – в максимальной степени экономить свой бюджет, а ограничения гарантируют, что каждый потребитель обслуживается не более чем одним предприятием производителя. Также из этих ограничений и определения целевой функции следует, что покупка совершается в том случае, когда это позволяет бюджет потребителя.

По аналогии с задачами LDP, LMP и LUP определяются задачи $MLMP$ и $MLUP$ по задаче $MLDP$. Введенное выше условие кооперативности позволяет нам говорить об оптимальном решении определенных выше задач размещения производства и ценообразования. Каждый потребитель выбирает в случае равенства затрат самое близкое предприятие относительно матрицы транспортных затрат, что позволяет производителю явным образом вычислить свой доход.

Перепишем двухуровневую задачу LDP в виде следующей задачи квадратичного программирования со смешанными переменными:

$$\sum_{i\in I}\sum_{j\in J} p_{ij}x_{ij} - \sum_{i\in I} f_i y_i \to \max_{p,x,y}$$

$$\sum_{i\in I}(b_j - c_{ij} - p_{ij})x_{ij} \geq 0; j \in J;$$

$$\sum_{i\in I}(c_{ij} + p_{ij})x_{ij} \leq c_{kj} + p_{kj}; k \in I, j \in J;$$

$$\sum_{i\in I} x_{ij} \leq 1; j \in J;$$

$$x_{ij} \leq y_i; i \in I, j \in J;$$

$$p_{ij} \geq 0; x_{ij}, y_i \in \{0,1\}; i \in I, j \in J.$$

Целевая функция определяет доход производителя. Первое ограничение гарантирует, что потребитель не выйдет за рамки своего бюджета. Выполнение второго ограничения приводит к тому, что транспортные затраты потребителя и его затраты на приобретение продукции в сумме минимальны. Третье ограничение означает, что каждый потребитель может быть обслужен не более чем в одном предприятии. Из четвертого ограничения следует, что потребитель может обслуживаться только открытым предприятием.

Также перепишем двухуровневую задачу $MLDP$ в виде следующей задачи квадратичного программирования со смешанными переменными:

$$\sum_{i \in I} \sum_{j \in J} p_{ij} x_{ij} \to \max_{p,x,y}$$

$$\sum_{i \in I} (b_j - c_{ij} - p_{ij}) x_{ij} \geq 0; j \in J;$$

$$\sum_{i \in I} (c_{ij} + p_{ij}) x_{ij} \leq (c_{kj} + p_{kj}) y_k; k \in I, j \in J;$$

$$\sum_{i \in I} y_i = r;$$

$$\sum_{i \in I} x_{ij} \leq 1; j \in J;$$

$$x_{ij} \leq y_i; i \in I, j \in J;$$

$$p_{ij} \geq 0; x_{ij}, y_i \in \{0,1\}; i \in I, j \in J.$$

Целевая функция, как и в двухуровневой задаче, определяет доход компании. Первая группа ограничений гарантируют, что потребители не могут выйти за рамки своих бюджетов. Выполнение второй группы

ограничений приводит к минимизации общих затрат каждого потребителя на покупку продукта и его транспортировку. Следующее ограничение гарантирует, что будет открыто ровно r предприятий. Четвертая группа ограничений означает, что каждый потребитель может обслуживаться не более чем на одном предприятии. Последняя группа ограничений влечёт, что потребители могут обслуживаться только открытыми предприятиями.

Сохраним для данных переформулировок двухуровневых задач те же обозначения LDP и $MLDP$. Аналогичным образом получим эквивалентные одноуровневые представления для двухуровневых задач размещения с равномерным и фабричным ценообразованием. Также будем использовать обозначения LMP, LUP, $MLMP$ и $MLUP$ для соответствующих одноуровневых переформулировок двухуровневых задач.

В дальнейшем предполагаем, что все исходные данные (f_i, b_j и c_{ij}) являются рациональными числами.

1.3 Задача конкурентного размещения и ценообразования

Задачи размещения это широкий и многообразный класс задач [35, 44, 46, 47, 48, 49, 50, 56, 72]. Среди них в последние годы большое внимание привлекают постановки, в которых решение о размещении принимают независимые игроки конкурирующие между собой. Исследования в области конкурентных задач размещения были начаты в [62], где рассматривался процесс выбора размещения предприятий и выбор политики ценообразования двумя конкурентами на конечном отрезке с рав-

номерным распределением покупателей. К настоящему моменту в этой области получено много интересных результатов. Поэтому ряд авторов значительные усилия посвятили классификации и обзору конкурентных моделей размещения. Существующие классификации основываются на разных интерпретациях процессов размещения [46, 47, 48, 49, 56, 72].

Важным элементом этих классификаций является способ представления пространства, где происходит размещение: n-мерное евклидово пространство, сеть и дискретное пространство. Иногда ограничиваются только первыми двумя пространствами [75]. Задачи размещения в любом из этих пространств делят на две части – непрерывные и дискретные задачи. В непрерывных постановках каждая точка сети или евклидова пространства являются возможным местом размещения производства. Дискретные задачи возникают в том случае, когда конечное множество возможных точек размещения считается известным заранее.

Ряд классификаций основывается на представлениях развитых в теории игр [48, 49, 52, 59, 65, 73, 78]. Конкуренты могут принимать решения одновременно или последовательно. Конкуренция может развиваться в пространстве и во времени, когда игроки по очереди меняют свои решения оценивая действия конкурента. Первые исследования в области последовательной конкуренции в задачах размещения, были проведены в работах [59, 73]. Отличительная черта этих постановок – наличие двух типов игроков. Первый тип – лидер, он принимает решение о размещении первым. Второй тип – конкурент, он принимает решение о размещении вторым, учитывая решение лидера. В этих задачах могут быть несколько лидеров и несколько конкурентов. Соответственно, в качестве концеп-

ции решения в этой области используется равновесие Штакельберга [78]. В постановках, в которых игроки принимают решения одновременно, как в [62], используется равновесие Нэша. Очень важное и интересное направление в этой области, связанное с теорией игр, это постановки, в которых конкурентное размещение связывается с ценообразованием [48, 49, 52, 65]. При этом обычно рассматриваются введенные ранее три стратегии ценообразования: фабричная, равномерная и дискриминационная.

Исследование в рамках одного контекста процессов ценообразования и размещения приводит к более сложным в вычислительном отношении задачам. Таким образом возникает необходимость в проведении исследований для разработки эффективных методов их решения. Ситуация ещё больше усложняется при исследовании процессов ценообразования и размещения в рамках двухуровневых моделей, которые вычислительно ещё сложнее, чем классические одноуровневые оптимизационные постановки [41]. Однако интерес к ним постоянно растёт, так как подобные постановки оказываются более адекватными в сравнении с одноуровневыми при моделировании изучаемых процессов в случае конкурирующих игроков, которые принимают решение последовательно [2, 18, 52, 65]. В связи с этим в работе предлагается такая постановка, в которой удается разделить процессы ценообразования и размещения, и таким образом получить более простую в вычислительном отношении модель. Идея такого разделения заключается в том, чтобы при фиксированных выборах размещения предприятий лидера и конкурента, использовать какую-нибудь модель ценовой конкуренции, которая приводит к равновесию Нэша [17].

Далее в этом качестве будет использоваться известная модель Бертрана [17].

Рассматриваемая в работе постановка хорошо описывается игрой Штакельберга, в которой первым выбирает размещение своих предприятий лидер, а затем конкурент. Далее игра развивается в соответствии со сценарием модели ценовой конкуренции Бертрана. Полученные равновесные цены используются для раздела рынков. Рынок достаётся тому из конкурентов, который может предложить наименьшую цену. На каждом из монополизированных рынков игрок устанавливает свою монопольную цену. Доход игрока складывается из доходов, которые он получает со своих монополизированных рынков.

Предлагаемая в работе новая модель конкурентного размещения производства и ценообразования формулируется в виде задачи двухуровневого линейного булевого программирования [41]. Для этой модели предлагаются приближённые алгоритмы на основе идей изложенных в [31]. Подход к моделированию процессов размещения и ценообразования реализованный в данной работе позволяет анализировать разные постановки. В частности, можно исследовать вопросы связанные с эластичностью спроса, эффектом каннибализации спроса, расширением рынка [18, 65, 71, 84, 85].

Перейдем к непосредственному описанию постановки. Рассмотрим следующую игру Штакельберга. Задано конечное множество пунктов размещения предприятий и конечное множество рынков. Предполагается, что предприятия производят однородный продукт. Первым ходом лидер размещает свои предприятия, затем свой выбор делает конкурент. У каждо-

го игрока имеются ограничения на бюджет для открытия предприятий. Для каждого рынка и предприятия известна общая себестоимость производства и доставки товара потребителям рынка. После того, как выбраны предприятия, процесс ценообразования на каждом рынке реализуется на основе модели ценовой конкуренции Бертрана. В этой модели игроки конкурируют между собой изменяя цены на продукцию стремясь к себестоимости производимой продукции. В этой работе как и в ряде других работ [71] идеология классической модели Бертрана используется для раздела рынков между игроками. В результате ценовой конкуренции рынки будут поделены между лидером и конкурентом. Предприятие лидера захватывает рынок, если себестоимость поставляемого им продукта минимальна среди всех открытых предприятий. Оставшиеся рынки захватываются предприятиями конкурента. Монополист на каждом своём рынке, в отличии от модели Бертрана, устанавливает оптимальную монопольную цену и получает прибыль равную произведению величины спроса на разность монопольной цены и себестоимости. Прибыль игрока складывается из прибыли с каждого из монополизированных им рынков. Цель игры лидера – выбрать такое множество пунктов размещения при заданном бюджетном ограничении, которое позволяет монополизировать рынки, доставляющие максимальную суммарную прибыль.

В качестве бюджетных ограничений далее используются ограничения медианного типа на количество выбранных предприятий лидера и конкурента, т.е. тип размещения II, описанный выше.

Введём обозначения:

$I = \{1, ..., n\}$ — множество возможных пунктов размещения для пред-

приятий лидера и конкурента;

$K = \{1, ..., m\}$ — множество рынков;

w — количество предприятий, размещаемых лидером;

r — количество предприятий, размещаемых конкурентом;

$c_i \geq 0$ — затраты на производство в пункте i;

$t_{ik} \geq 0$ — транспортные затраты для перевозки продукции из пункта i на рынок k;

$c_{ik} = c_i + t_{ik}$ — себестоимость обслуживания k-го рынка из пункта i.

Для описания выбора лидера и конкурента будем использовать следующие переменные:

$$x_i = \begin{cases} 1, & \text{если лидер размещает в пункте } i \text{ свое предприятие,} \\ 0 & \text{в противном случае.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{если конкурент размещает в пункте } i \text{ свое предприятие,} \\ 0 & \text{в противном случае.} \end{cases}$$

Если вектор x – выбор лидера, а вектор y – выбор конкурента, то обозначим через $c_k(x) = \min\{c_{ik} \mid x_i = 1\}$ минимальную себестоимость обслуживания рынка k предприятиями лидера, соответственно, через $c_k(y) = \min\{c_{ik} \mid y_i = 1\}$ обозначим минимальную себестоимость обслуживания рынка k предприятиями конкурента.

Когда места размещения предприятий у лидера и у конкурента выбраны, то на каждом рынке между игроками начинается ценовая конкуренция. Она реализуется на основе модели Бертрана. Поэтому на любом рынке побеждает тот из соперников, у которого наименьшая себестоимость обслуживания данного рынка. Однако, в отличии от классической модели ценовой конкуренции Бертрана, в которой эта наименьшая себе-

стоимость и выбирается в качестве цены товара, в этой работе используется более реалистическая гипотеза о цене товара. После того как рынки распределены между конкурентами начинается процесс определения монопольных цен и прибыли на каждом их них.

В данной работе мы рассмотрим простой случай определения монопольной цены. Пусть рынок k монополизирован лидером. Будем считать, что максимальный доход лидера на этом рынке задаётся функцией $\varphi_k(c_k(x))$, где $\varphi_k(c)$ – монотонно невозрастающая функция аргумента c. Аналогично определим максимальный доход для конкурента $\varphi_k(c_k(y))$, если данный рынок монополизирован конкурентом.

Предложенный механизм конкурентного ценообразования, вычисления монопольных цен и прибыли позволяет обойтись в математической модели, как покажет дальнейшее, без ценовых переменных. Введённые ранее обозначения, соглашения и полученные выше формулы позволяют записать игру Штакельберга в виде задачи двухуровневого линейного булевого программирования.

Пусть множество $I_k(x)$ состоит из пунктов размещения, которые может использовать конкурент для захвата k-го рынка:

$$I_k(x) = \{i : c_{ik} < \min_{l:x_l=1} c_{lk}\}.$$

Обозначим через $\overline{C} = \max_{ik} c_{ik}$ – максимально возможную себестоимость обслуживания рынков.

Введём дополнительные переменные:

$$z_{ik}^L = \begin{cases} 1, & \text{если } k\text{-й рынок обслуживается } i\text{-м предприятием лидера,} \\ 0 & \text{в противном случае;} \end{cases}$$

$$z_{ik}^F = \begin{cases} 1, & \text{если } k\text{-й рынок обслуживает } i\text{-е предприятие конкурента,} \\ 0 & \text{в противном случае.} \end{cases}$$

Пусть $d_{ik} = \varphi_k(c_k(x))$, что содержательно интерпретируется как доход, получаемый монополистом от k-го рынка, если минимальная себестоимость обслуживания рынка предприятиями монополиста $c_k(x) = c_{ik}$. Используя переменные z^L, z^F и введённые ранее переменные x, y получим следующую задачу двухуровневого программирования CLP (Competitive Location and Pricing problem):

$$\sum_{k \in K} \sum_{i \in I} d_{ik} z_{ik}^L \to \max_{x, y, z^L, z^F} \tag{1.6}$$

при ограничениях:

$$\sum_{i \in I} x_i = w; \tag{1.7}$$

$$(y, z^L, z^F) \in F^*(x); \tag{1.8}$$

$$x_i \in \{0, 1\}; i \in I, \tag{1.9}$$

Целевая функция (1.6) определяет доход лидера, который складывается из его доходов от монополизированных рынков. Ограничение (1.7) означает, что лидер должен открыть ровно w предприятий. Выполнение ограничения (1.8) приводит к тому, что раздел рынков и выбор монопольных цен, определяющих доходы игроков, осуществляется с учётом оптимальной реакции конкурента. Именно это ограничение и превращает данную постановку в задачу двухуровневого программирования. Множество $F^*(x)$ является множеством оптимальных решений параметрической задачи конкурента. В качестве параметра выступает множество выбранных лидером пунктов производства.

Задача конкурента $F(x)$ выглядит следующим образом:

$$\sum_{k\in K}\sum_{i\in I} d_{ik} z^F_{ik} \to \max_{y,z^L,z^F} \tag{1.10}$$

при ограничениях:

$$\sum_{i\in I} y_i = r; \tag{1.11}$$

$$\sum_{i\in I} z^F_{ik} \le \sum_{i\in I_k(x)} y_i; k \in K; \tag{1.12}$$

$$x_i + y_i \le 1; i \in I; \tag{1.13}$$

$$\sum_{i\in I}(z^L_{ik} + z^F_{ik}) = 1; k \in K; \tag{1.14}$$

$$z^L_{ik} \le x_i; i \in I, k \in K; \tag{1.15}$$

$$c_{ik} z^L_{ik} \le c_{i'k} x_{i'} + (1 - x_{i'})\overline{C}; i \in I, k \in K, i' \in I; \tag{1.16}$$

$$z^F_{ik} \le y_i; i \in I, k \in K; \tag{1.17}$$

$$c_{ik} z^F_{ik} \le c_{i'k} y_{i'} + (1 - y_{i'})\overline{C}; i \in I, k \in K, i' \in I; \tag{1.18}$$

$$z^L_{ik}, z^F_{ik}, y_i \in \{0, 1\}; i \in I, k \in K. \tag{1.19}$$

Целевая функция (1.10) определяет доход конкурента, который складывается из его доходов от монополизированных рынков. Ограничение (1.11) означает, что конкурент должен открыть ровно r предприятий. Ограничения (1.12) и (1.14) реализуют механизм раздела рынков. Если $I_k(x) = \emptyset$, то лидер захватывает рынок, так как у него минимальная себестоимость обслуживания рынка. В противном случае рынок может оказаться у конкурента, если он выберет один из пунктов множества $I_k(x)$ в качестве места размещения одного из своих предприятий. Ограничение (1.13) запрещает лидеру и конкуренту размещать предприятия в одном и том же месте. Ограничения (1.14) – (1.17) гарантируют, что

лидер монополизирует рынок k только в том случае, когда минимальная себестоимость продукции, предоставляемая его предприятиями, не превосходит минимальной себестоимости, которую может достигнуть на этом рынке конкурент. Из этих ограничений также следует, что выбрана единственная пара открытых предприятий, одно лидера, а другое конкурента. Причём на выбранном предприятии лидера достигается минимальная себестоимость продукции на данном рынке. Точно также и для конкурента на выбранном предприятии достигается минимальная себестоимость продукции на данном рынке. Аналогичным образом интерпретируются ограничения (1.13), (1.14), (1.17) и (1.18) для рынка k, монополизированного конкурентом.

1.4 Задача государственно-частного партнерства

В настоящей работе изучается интересная и важная с прикладной точки зрения задача формирования работоспособного механизма государственно-частного партнерства, которая формулируется в виде задачи двухуровневого булевого программирования и основывается на идеях Хотеллинга и механизме дисконтирования доходов и расходов, полученных и произведенных в разные моменты времени [62]. Решение такой задачи позволяет построить эффективную программу освоения минерально-сырьевой базы региона и может быть использовано для поддержки процесса принятия управленческих решений в природно-ресурсной сфере [10, 11].

В работах [10, 11] предложена концепция формирования программы освоения минерально-сырьевой базы на основе механизмов государст-

венно-частного партнёрства, согласующих долгосрочные интересы государства, частного инвестора и населения в процессе социально-экономического развития и обеспечивающих инвестиционную привлекательность, бюджетные поступления, соблюдение экологических ограничений и рост индикаторов уровня жизни. В рамках данной концепции в процессе освоения территории государство берет на себя не только реализацию инфраструктурных проектов общего назначения, но и часть затрат, связанных с компенсацией экологических потерь, вызванных реализацией инвестиционных проектов.

На вход рассматриваемой далее модели планирования подаётся следующий перечень данных:

- набор производственных (инвестиционных) проектов, реализуемых частным инвестором, конкретную конфигурацию которых инвестор выбирает в зависимости от того, что предлагает государство в области инфраструктурного и экологического строительства;
- набор инфраструктурных проектов, реализуемых государством, конкретный перечень которых оно выбирает, исходя из своих оценок эффективности с точки зрения перспектив долгосрочного развития территории;
- перечень экологических проектов, необходимых для компенсации экологических потерь, вызванных реализацией инвестиционных и инфраструктурных проектов.

Выход модели – программа развития территории (набор реализуемых проектов) и механизм раздела затрат в процессе реализации ин-

фраструктурных и экологических проектов между государством и инвестором.

Формальное описание задачи планирования может быть представлено следующим образом. Введём обозначения:

I – множество производственных (инвестиционных) проектов;

J – множество инфраструктурных проектов;

K – множество экологических проектов;

T – горизонт планирования.

Инвестиционные проекты:

CFP_i^t – поток наличности инвестиционного проекта i в год t (кэшфло; равен разности полученных доходов и расходов всех видов);

EPP_i^t – стоимостная оценка экологических потерь при реализации проекта i в год t;

DBP_i^t – доходы бюджета от реализации проекта i в год t;

ZPP_i^t – зарплата, выплачиваемая в ходе реализации проекта i в год t.

Инфраструктурные проекты:

ZI_j^t – график затрат на реализацию проекта j в год t;

EPI_j^t – стоимостная оценка экологических потерь при реализации проекта j в год t;

VDI_j^t – внепроектные доходы бюджета от реализации проекта j, связанные с общим развитием экономики территории, в год t;

ZPI_j^t – зарплата, выплачиваемая в ходе реализации проекта j в год t.

Экологические проекты:

ZE_k^t – график затрат на реализацию проекта k в год t;

EDE_k^t – стоимостная оценка экологического дохода при реализации проекта k в год t;

ZPE_k^t – зарплата, выплачиваемая в ходе реализации проекта k в год t.

Взаимосвязь проектов задаётся величинами μ и ν, где μ_{ij} – индикатор технологической связности производственных и инфраструктурных проектов, $i \in I$, $j \in J$, а ν_{ij} – индикатор связности производственных и экологических проектов, $i \in I$, $k \in K$:

$$\mu_{ij} = \begin{cases} 1, & \text{если для реализации производственного проекта } i \\ & \text{требуется реализация инфраструктурного проекта } j \\ 0 & \text{в противном случае;} \end{cases},$$

$$\nu_{ik} = \begin{cases} 1, & \text{если для реализации производственного проекта } i \\ & \text{требуется реализация экологического проекта } k \\ 0 & \text{в противном случае.} \end{cases},$$

Дисконты и бюджетные ограничения:

DG – дисконт государства;

DI – дисконт инвестора;

b_t^G – бюджет государства в год t;

b_t^O – бюджет инвестора в год t.

Для описания выбора государства и инвестора будем использовать следующие переменные:

$$x_j = \begin{cases} 1, & \text{если государство запускает } j\text{-й инфраструктурный проект,} \\ 0 & \text{в противном случае;} \end{cases}$$

$$y_k = \begin{cases} 1, & \text{если государство запускает } k\text{-й экологический проект,} \\ 0 & \text{в противном случае;} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{если инвестор запускает } i\text{-й производственный проект,} \\ 0 & \text{в противном случае;} \end{cases}$$

$$u_k = \begin{cases} 1, & \text{если инвестор запускает } k\text{-й экологический проект,} \\ 0 & \text{в противном случае.} \end{cases}$$

Задача государства (Public-Private Partnership) PPP:

$$\sum_{t\in T}\Big(\sum_{i\in I}(DBP_i^t + ZPP_i^t - EPP_i^t)z_i + \sum_{j\in J}(VDI_j^t + ZPI_j^t - EPI_j^t - ZI_j^t)x_j +$$

$$+\sum_{k\in K}(EDE_k^t + ZPE_k^t - ZE_k^t)y_k + \sum_{k\in K}(EDE_k^t + ZPE_k^t)u_k\Big)/(1+DG)^t \to$$

$$\to \max_{x,y,z,u} \tag{1.20}$$

при ограничениях:

$$\sum_{j\in J} ZI_j^t x_j + \sum_{k\in K} ZE_k^t y_k \le b_t^G; t \in T; \tag{1.21}$$

$$(z,u) \in \mathcal{F}_{PI}^*(x,y); \tag{1.22}$$

$$x_j, y_k \in \{0,1\}; j \in J, k \in K. \tag{1.23}$$

Целевая функция (1.20) определяет чистый дисконтированный доход государства. В год t доход государства складывается из трёх составляющих. Первая из них образована доходами бюджета от реализации производственных проектов инвестором и зарплатой, выплачиваемой в процессе их реализации, за вычетом затрат, определяемых экологическими потерями, сопутствующими текущему выбору инвестора. Вторая составляющая образована внепроектными доходами бюджета от реализации

выбранных государством инфраструктурных проектов, зарплатой, выплачиваемой в процессе их осуществления, затратами на их реализацию и экологическими потерями. Третья группа доходов и расходов связана с экологическими проектами. Бюджетные ограничения (1.21) отражают затраты государства на реализацию выбранных инфраструктурных и экологических проектов. Выполнение ограничения (1.22) приводит к тому, что механизм раздела затрат в процессе реализации инфраструктурных и экологических проектов между государством и инвестором учитывает оптимальную реакцию инвестора. Именно это ограничение и превращает данную постановку в задачу двухуровневого программирования. Множество $\mathcal{F}^*_{PI}(x, y)$ является множеством оптимальных решений параметрической задачи инвестора. В качестве параметров выступает множество выбранных государством инфраструктурных проектов x и экологических проектов y.

Задача инвестора $PI(x, y)$:

$$\sum_{t\in T}\Big(\sum_{i\in I} CFP_i^t z_i - \sum_{k\in K} ZE_k^t u_k\Big)/(1+DI)^t \to \max_{z,u} \tag{1.24}$$

при ограничениях:

$$\sum_{k\in K} ZE_k^t u_k - \sum_{i\in I} CFP_i^t z_i \le b_t^O; t \in T; \tag{1.25}$$

$$\sum_{t\in T}\Big(\sum_{i\in I}(ZPP_i^t - EPP_i^t)z_i + \sum_{j\in J}(ZPI_j^t - EPI_j^t)x_j + \\ + \sum_{k\in K}(EDE_k^t + ZPE_k^t)(y_k + u_k)\Big)/(1+DI)^t \ge 0; \tag{1.26}$$

$$\sum_{t\in T}\Big(\sum_{i\in I} CFP_i^t z_i - \sum_{k\in K} ZE_k^t u_k\Big)/(1+DI)^t \ge 0; \tag{1.27}$$

$$x_j \ge \mu_{ij} z_i; i \in I, j \in J; \tag{1.28}$$

$$y_k + u_k \leq 1; k \in K; \tag{1.29}$$

$$y_k + u_k \geq \nu_{ik} z_i; i \in I, k \in K; \tag{1.30}$$

$$z_i, u_k \in \{0, 1\}; i \in I, k \in K. \tag{1.31}$$

Целевая функция (1.24) определяет чистый дисконтированный доход инвестора. В год t его доход определяется потоком наличности реализуемых им инвестиционных проектов и затратами на реализацию выбранных экологических проектов. Из (1.25) следует, что затраты инвестора на реализацию экологических проектов в год t не превосходят текущий бюджет и кэшфло реализуемых в этот год инвестиционных проектов. Ограничение (1.26) является формальным аналогом референдума населения. Его выполнение говорит о том, что достигнут компромисс между интересами государства, населения и инвестора. Ограничение (1.27) гарантирует, что механизм раздела затрат в процессе реализации инфраструктурных и экологических проектов между государством и инвестором обеспечивает возможность получения инвестором нормальной прибыли. Из ограничения (1.28) следует, что если $\mu_{ij} = 1$, то i-й инвестиционный проект может быть реализован только при условии, что государством реализован j-й инфраструктурный проект. Ограничения (1.29) гарантируют, что k-й экологический проект может быть реализован только одним из участников партнерства. Из ограничения (1.30) следует, что если $\nu_{ik} = 1$, то i-й инвестиционный проект может быть реализован только при условии, что либо государством, либо инвестором реализован k-й экологический проект.

Содержательно двухуровневую постановку (1.20) – (1.23) можно интерпретировать с помощью подходящей игры Штакельберга, в которой

участвуют два игрока: государство и инвестор. Первым решение принимает государство, а затем – инвестор. Под допустимым решением задачи (1.20 – 1.23) понимается любой вектор (x, y, z, u), который удовлетворяет условиям (1.21) – (1.23). Традиционно в задачах двухуровневого программирования используются два определения понятия "наилучшего" решения: оптимальные (оптимистические) решения и наилучшие гарантированные (пессимистические) решения [41]. Иногда также говорят, соответственно, кооперативные и некооперативные решения (особенно в тех случаях, когда необходимо подчеркнуть игровую составляющую моделируемой ситуации, т.е. кооперируются игроки или нет в процессе принятия решения). Необходимость в фиксации такого рода отличий не возникает, если, например, задача инвестора имеет единственное оптимальное решение. В общем случае, приходится уточнять как игроки взаимодействуют между собой. Как видно из постановки (1.20) – (1.23), в работе рассматривается ситуация, когда инвестор кооперируется с государством. В данной ситуации это естественное соглашение, т.к. государство полностью берёт на себя выполнение затратных инфраструктурных проектов, требующих большого времени для окупаемости. Поэтому для инвестора нет мотивов как-то конкурировать с государством или стремиться нанести ему какой-либо ущерб. Хотя с теоретической точки зрения, интересно и полезно оценить ущерб, который возникает в случае некооперативного поведения инвестора.

Глава 2

Задачи ценообразования

В этой главе показано, что рассматриваемая задача фабричного ценообразования, в отличии от задач равномерного и дискриминационного ценообразования относится, как и многие другие интересные задачи комбинаторной оптимизации, к классу труднорешаемых задач. При разработке методов их решения возникает следующая дилемма. Если сосредоточиться на построении точных методов, то, как показывает вычислительная практика, такие подходы становятся несостоятельными из-за нехватки ресурсов. С другой стороны, современный опыт построения метаэвристик показывает, что для большинства задач удаётся построить малотрудоёмкие приближённые алгоритмы решения [13, 15, 22, 23, 74]. Однако возникает проблема с оценкой качества таких решений. Рассматриваемые во второй главе гибридные алгоритмы – попытка совместить сильные стороны того и другого направлений решения NP-трудных задач. Последние десятилетия это направление, лежащее на стыке математического программирования, комбинаторной оптимизации, математической логики и программирования быстро развивается [32, 39, 60, 61, 66, 67, 80]. Гибридизация методов глобального поиска и метаэвристик – одно из

главных направлений проводимых исследований [60].

Предлагаемые в этой главе алгоритмы для решения задачи фабричного ценообразования основываются на использовании декомпозиции и метаэвристик. Такой подход к разработке точных и приближенных алгоритмов использовался в работе [22]. Проведенный там эксперимент показал работоспособность разработанных точных и приближенных алгоритмов для задачи о (r|p)-центроиде. В диссертационной работе оценивается эффективность данных гибридных схем при поиске точных и приближенных решений для задач имеющих "компактное" представление допустимой области в отличии от экспоненциального по числу ограничений представления, которое использовалось при описании задачи о (r|p)-центроиде в виде смешано-целочисленной задачи.

Основной недостаток декомпозиционных алгоритмов – медленная сходимость [69]. Поэтому внимание также уделялось поиску методов ускорения декомпозиционных алгоритмов. Ранее подобные исследования проводились, например, в [54, 68, 69, 82]. В [14, 22] эта проблема рассматривалась с точки зрения применения метаэвристик.

В дополнение, во второй главе диссертации устанавливается принадлежность задачи фабричного ценообразования к классу Log-APX, а также показано, что она не может быть аппроксимируемой с абсолютной погрешностью ограниченной константой при условии несовпадения классов P и NP.

2.1 Необходимые условия оптимальности. Полиномиально разрешимые случаи задачи фабричного ценообразования

В данном разделе вводятся необходимые условия оптимальности, которые используются при анализе сложности задачи фабричного ценообразования и при разработке одного из точных алгоритмов её решения. Как будет показано в разделе 2.2, задача MP является NP-трудной. Поэтому ниже приводятся ее полиномиально разрешимые случаи, вытекающие из точных алгоритмов ее решения и необходимых условий оптимальности.

Пусть (p, x) – произвольное допустимое решение задачи MP. Введем следующие обозначения:

$I_s := \{i \in I | \exists j \in J : x_{ij} = 1, b_j - c_{ij} - p_i = 0\}$;

$J_i := \{j \in J | x_{ij} = 1\}, i \in I$;

$I_j := \{i \in I | c_{ij} + p_i = \min\limits_{k \in I} \{c_{kj} + p_k\}\}, j \in J$;

$g_{ik} := 0$, если $\exists j \in J : i, k \in I_j$, и $g_{ik} := 1$, в противном случае;

$\tilde{I} := \{i \in I | \sum\limits_{j \in J} x_{ij} \geqslant 1\}$.

Будем говорить, что $i, k \in I$ принадлежат одному классу зависимости I_d тогда и только тогда, когда найдутся такие $\alpha_1, ..., \alpha_r, r \leqslant n - 2$, что:

$$g_{i\alpha_1} = g_{\alpha_1\alpha_2} = ... = g_{\alpha_r k} = 0.$$

Теорема 1 (Необходимые условия оптимальности) *Если (p, x) — произвольное оптимальное решение задачи MP, то:*

1)для любого $i \in \tilde{I}$ либо $i \in I_s$, либо найдется номер $j \in J_i$ такой, что $|I_j| > 1$;

2)для любого I_d, если $I_d \cap \tilde{I} \neq \emptyset$, то $I_d \cap I_s \neq \emptyset$.

Доказательство ▷ 1) Допустим обратное. Пусть существует $i \in \tilde{I}$ такое, что $i \notin I_s$ и для любого $j \in J$ имеем, что $|I_j| = 1$. Тогда для любого $j \in J_i$ выполняется неравенство $b_j - c_{ij} - p_i > 0$ и любой столбец с номером $j \in J_i$ содержит ровно один минимум, то есть для любого $k \neq i$ имеем $c_{ij} + p_i < c_{kj} + p_k$. Отсюда следует, что можно увеличить переменную p_i так, что не нарушатся ограничения (1.2) и (1.3) и значение целевой функции увеличится. Действительно, положим:

$$\varepsilon_1 = \min_{j \in J_i} \{b_j - c_{ij} - p_i\};$$
$$\varepsilon_2 = \min_{j \in J_i} \min_{k \neq i} \{c_{kj} + p_k - c_{ij} - p_i\};$$
$$\varepsilon = \min\{\varepsilon_1, \varepsilon_2\}.$$

Увеличим i-ую компоненту вектора p на ε. И получим новый вектор цен $\tilde{p}$. Из определения ε следует, что ограничения (1.2) и (1.3) не нарушаются, а значение целевой функции на новом допустимом решении $(\tilde{p}, x)$ увеличится на $\varepsilon|J_i|$. А это противоречит оптимальности вектора (p, x). Таким образом для любого $i \in \tilde{I}$ либо $i \in I_s$, либо найдется номер $j \in J_i$ такой, что $|I_j| > 1$.

2) Из определения I_d следует, что каждое из них состоит из объединения множеств I_j и, если I_d и I'_d два разных подмножества, то они не пересекаются. Отсюда следует, что если существует множество I_d, для которого $I_d \cap \tilde{I} \neq \emptyset$ и $I_d \cap I_s = \emptyset$, то как и в первой части утверждения можно подправить цены так, что увеличится значение целевой функции. Действительно, рассмотрим величины: $\varepsilon_1 = \min_{I_j \subseteq I_d} \min_{k \in I \setminus I_j} \{c_{kj} + p_k - d_j\}$, где $d_j = \min_{i \in I_j} \{c_{ij} + p_i\}$, если множество $I \setminus I_j$ пусто, то считаем, что внутренний минимум равен достаточно большому положительному числу; $\varepsilon_2 = \min_{i \in I_d} \min_{j \in J_i} \{b_j - c_{ij} - p_i\}$.

В силу предположения $I_d \cap I_s = \emptyset$ и определения величин ε_1 и ε_2 получим, что их минимум будет положительным, то есть $\varepsilon = \min\{\varepsilon_1, \varepsilon_2\} > 0$. Увеличим компоненты вектора p с номерами из множества I_d на величину ε. Получим новый вектор цен $\tilde{p}$. Из определения ε следует, что ограничения (1.2) и (1.3) не нарушаются, а значение целевой функции на новом допустимом решении $(\tilde{p}, x)$ увеличится за счет предприятия $i \in I_d \cap \tilde{I}$. А это противоречит оптимальности вектора (p, x). Таким образом предположение неверно и выполняется требуемое свойство $I_d \cap I_s \neq \emptyset$. Теорема доказана. $\triangleleft$

Опираясь на необходимые условия оптимальности, приведём два точных алгоритма решения задачи, которые имеют полиномиальную трудоёмкость либо при фиксированном числе клиентов, либо при фиксированном количестве предприятий.

Из вида ограничений задачи MP следует, что для заданного вектора цен или для заданной матрицы назначений задача станет полиномиально разрешимой. В первом случае необходимо вычислить 0-1 матрицу назначений, удовлетворяющую ограничениям (1.2) – (1.5). Заметим, что для некоторого предприятия $i \in I$ и потребителя $j \in J$ выполнено $x_{ij} = 1$ тогда и только тогда, когда для всех $k \neq i$ имеем $x_{kj} = 0$, к тому же сумма затрат j-го потребителя минимальна на i-ом предприятии, при том в случае равенства затрат на предприятиях i и k выполнено $p_i \geq p_k$, и затраты j-го потребителя не превышают его бюджет. То есть, для поиска матрицы назначений каждому потребителю необходимо сопоставить предприятие с минимальными затратами, не превышающими бюджет, а в случае равенства затрат выбирается предприятие с максимальной ценой.

Получается, что задача MP с фиксированным вектором цен решается за $O(nm)$. Также не трудно убедиться, что задача MP с фиксированной матрицей назначений является задачей линейного программирования и, следовательно, полиномиально разрешима [16]. Если использовать алгоритм Кармаркара, то для решения задачи потребуется $O(n^{3,5}L)$ арифметических операций при условии, что величины $b_j, c_{ij}, i \in I, j \in J$, являются $O(L)$-разрядными числами [16]. Исходя из этих соображений, для нахождения оптимального решения задачи MP можно воспользоваться одним из двух следующих переборных алгоритмов:

Алгоритм 1 (Перебор назначений) *Обозначим за X – множество всех 0-1 матриц размера $n \times m$, удовлетворяющих ограничению (1.4). $\widetilde{X} \subseteq X$.*

Шаг 0 *Положим $\widetilde{X} := \emptyset$, $f^* := 0$;*

Шаг 1 *Выберем матрицу $\widetilde{x} \in X \backslash \widetilde{X}$. С помощью алгоритма Кармаркара найдем оптимальный вектор цен $\widetilde{p}$ в задаче MP с фиксированной матрицей назначений $\widetilde{x}$;*

Шаг 2 *Если $f(\widetilde{p}, \widetilde{x}) > f^*$, тогда положим $p^* := \widetilde{p}$, $x^* := \widetilde{x}$, $f^* := f(p^*, x^*)$. $\widetilde{X} := \widetilde{X} \cup \widetilde{x}$. Если $X \setminus \widetilde{X} \neq \emptyset$, тогда переходим на* ***Шаг 1****. Иначе* ***СТОП****.*

Очевидно, что алгоритм 1 конечен, т.к. конечно множество X. Второй алгоритм основан на переборе неотрицательных векторов цен, которых, однако, континуальное количество. Поэтому, прежде чем описывать сам алгоритм, покажем, что для решения задачи MP достаточно обойтись перебором конечного числа векторов цен. Докажем следующие вспомогательные утверждения.

Лемма 1 (Лемма о дискретизации цен) *Если* (p, x) *– оптимальное решение задачи* MP*, то оптимальная цена* p_i *на любом предприятии* $i \in \tilde{I}$ *представляется в виде суммы* $b_{j_r} - \sum_{s=2}^{r}(c_{i_{s-1}j_s} - c_{i_{s-1}j_{s-1}}) - c_{ij_1}$*, для некоторого* $1 \leq r \leq n$ *и некоторых наборов из* r *потребителей и* $r-1$ *попарно различных предприятий, отличных от предприятия* i*.*

Доказательство ▷ Пусть известно некоторое оптимальное решение (p, x). По определению множества I_s, если предприятие $i \in I_s$, то найдется такой потребитель $j \in J$, что $b_j - c_{ij} = p_i$. Из теоремы 1 следует, что при условии $i \in \tilde{I} \setminus I_s$ найдется множество I_d, содержащее предприятие i, причем $I_d \cap I_s \neq \emptyset$, или другими словами, существует такой набор $(i = i_1, ..., i_{r-1})$ предприятий этого множества, что $g_{ii_1} = g_{i_1i_2} = g_{i_2i_3} = ... = g_{i_{r-1}i_r} = 0$, $i_r \in I_d \cap I_s$. По определению, существует обслуживаемый потребитель j_r, который тратит в i_r-ом предприятии весь свой бюджет, а также существуют такие клиенты $j_1, ..., j_{r-1}$, что $i, i_1 \in I_{j_1}; i_1, i_2 \in I_{j_2}; ...; i_{r-2}, i_{r-1} \in I_{j_{r-1}}$. Тогда оптимальная цена на предприятии i есть в точности $b_{j_r} - \sum_{s=2}^{r}(c_{i_{s-1}j_s} - c_{i_{s-1}j_{s-1}}) - c_{ij_1}$. ◁

В дополнение к лемме стоит отметить, что на любом предприятии $i \in \tilde{I}$ оптимальная цена $p_i \in [0, \max_{j \in J}(b_j - c_{ij})]$. Определим для каждого предприятия i множество цен $Price(i)$ следующим образом. Будем считать, что любая цена p_i из отрезка $[0, \max_{j \in J}(b_j - c_{ij})]$ принадлежит $Price(i)$ тогда и только тогда, когда $p_i = b_{j_r} - \sum_{s=2}^{r}(c_{i_{s-1}j_s} - c_{i_{s-1}j_{s-1}}) - c_{ij_1}$, для некоторого $1 \leq r \leq n$ и некоторых наборов из r потребителей и $r-1$ попарно различных предприятий, отличных от предприятия i. По лемме 1, множество $Price(i)$ включает все возможные значения i-й координаты оптимальных векторов цен при условии $i \in \tilde{I}$. Пусть

$Price = \prod_{i \in I} Price(i)$ – декартово произведение множеств $Price(i)$.

Теорема 2 (Теорема о дискретизации) *Существует оптимальное решение (p, x) задачи MP такое, что $p \in Price$.*

Доказательство ▷ Пусть (p, x) – произвольное оптимальное решение. По лемме 1 для всех $i \in \tilde{I}$ выполнено $p_i \in Price(i)$. Определим вектор цен $\tilde{p}$ следующим образом. Пусть $\tilde{p}_i = p_i$ для всех $i \in \tilde{I}$, а для всех $i \notin \tilde{I}$ за $\tilde{p}_i$ обозначим максимальный элемент $Price(i)$. Убедимся, что решение $(\tilde{p}, x)$ также оптимальное решение. По построению нового вектора цен решение $(\tilde{p}, x)$ допустимо, если выполнено ограничение (1.3). Пусть для некоторых $k \in I$ и $j \in J$ выполнено $\sum_{i \in I}(c_{ij} + \tilde{p}_i)x_{ij} > c_{kj} + \tilde{p}_k$. По построению вектора $\tilde{p}$ это возможно только в том случае, если $k \in I \setminus \tilde{I}$. Из определения множества $Price(k)$ следует, что $(b_j - c_{kj}) \in Price(k)$. Получается, что $\tilde{p}_k < \sum_{i \in I}(c_{ij} + \tilde{p}_i)x_{ij} - c_{kj} \leq (b_j - c_{kj}) \in Price(k)$ – противоречие. ◁

Из теоремы 1 следует, что множество $Price$ содержит оптимальный вектор цен. Тогда для поиска оптимального решения задачи MP можно использовать следующий конечный переборный алгоритм:

Алгоритм 2 (Перебор цен) *Пусть $P \subseteq Price$.*

Шаг 0 *Положим $P := \emptyset$, $f^* := 0$;*

Шаг 1 *Выберем вектор $\widetilde{p} \in Price \setminus P$. Найдем оптимальную матрицу назначений $\widetilde{x}$ в задаче MP с фиксированным вектором цен $\widetilde{p}$;*

Шаг 2 *Если $f(\widetilde{p}, \widetilde{x}) > f^*$, тогда положим $p^* := \widetilde{p}$, $x^* := \widetilde{x}$, $f^* := f(p^*, x^*)$. $P := P \cup \widetilde{p}$. Если $Price \setminus P \neq \emptyset$, тогда переходим на **Шаг 1**. Иначе **СТОП**.*

Приведем оценки трудоёмкости алгоритмов 1 и 2.

Теорема 3 *Имеют место следующие утверждения:*

1) Время работы алгоритма 1 оценивается величиной $O(n^{m+3,5}L)$, *при условии, что параметры задачи* $b_j, c_{ij}, i \in I, j \in J$, *являются* $O(L)$-*разрядными числами.*

2) Время работы алгоритма 2 имеет следующий порядок $O(nm(\sum\limits_{k=0}^{n-1} \frac{(n-1)!}{(n-k-1)!} m^k)^n)$.

Доказательство ▷ 1) Первое утверждение следует из того, что на каждом шаге алгоритма решается задача линейного программирования. Если использовать алгоритм Кармаркара, то для решения задачи потребуется $O(n^{3,5}L)$ арифметических операций при условии, что величины $b_j, c_{ij}, i \in I, j \in J$, являются $O(L)$-разрядными числами [16]. Т.к. 0-1 матриц размера $n \times m$, удовлетворяющих ограничению (1.4) не более $O(n^m)$, то умножив эту величину на трудоёмкость алгоритма Кармаркара получим требуемую оценку.

2) По определению, каждый элемент множества $Price(i)$ определяется упорядоченным набором из k предприятий, для некоторого $0 \leq k \leq n-1$, и предприятием i, при этом каждому из них сопоставляется некоторый клиент. Количество упорядоченных выборок размера k из $(n-1)$–элементного множества есть в точности $A_{n-1}^k = \frac{(n-1)!}{(n-k-1)!}$. Получаем $|Price(i)| = \sum\limits_{k=0}^{n-1} \frac{(n-1)!}{(n-k-1)!} m^k$. Тогда $|Price| = O((\sum\limits_{k=0}^{n-1} \frac{(n-1)!}{(n-k-1)!} m^k)^n)$. Так как задача MP с фиксированным вектором цен решается за $O(nm)$, то время работы алгоритма 2 оценивается величиной $O(nm(\sum\limits_{k=0}^{n-1} \frac{(n-1)!}{(n-k-1)!} m^k)^n)$.

◁

Теорема 4 *Задача MP полиномиально разрешима при выполнении любого из следующих условий:*

1) $|J| = m = const$*;*

2) $|I| = n = const$*.*

Доказательство ▷ Полиномиальная разрешимость задачи при условии $|J| = m = const$ следует из утверждения 1 теоремы 3. Аналогично, из утверждения 2 той же теоремы следует, что с помощью алгоритма 2 можно решить задачу MP за время

$$T_2(n, m) = O(nm(\sum_{k=0}^{n-1} \frac{(n-1)!}{(n-k-1)!} m^k)^n) \leq O(m^{n^2}).$$

Откуда и следует полиномиальная разрешимость задачи при условии $|I| = n = const$. ◁

2.2 Вычислительная сложность

Данный раздел посвящен анализу сложностного статуса задач MP, DP и UP. Показано, что задача MP является NP-трудной в сильном смысле, а задачи DP и UP полиномиально разрешимы. Устанавливается положение задачи MP в иерархии аппроксимационных классов.

Сперва покажем, что задачи DP и UP полиномиально разрешимы. Очевидно, что в задаче DP набор цен $\tilde{p}_{ij} = \max\{0, b_j - c_{ij}\}$ является оптимальным, т.е. существует оптимальное решение (x, p), где $p = \tilde{p}$. Подбирая для каждого клиента максимальную цену, которую он может потратить, мы найдем назначение x. Следовательно задачу DP можно решить за nm. Как и в теореме о необходимых условиях оптимальности задачи фабричного ценообразования в задаче UP оптимальная цена $p =$

$b_j - c_{ij}$, для некоторого $j \in J$. Следовательно, перебором по все ценам такого вида и подбором подходящего назначения можно решить задачу UP за n^2m^2.

Имеет место следующее утверждение.

Теорема 5 *Задача MP является NP-трудной в сильном смысле.*

Доказательство ▷ Рассмотрим NP-трудную в сильном смысле задачу о минимальном покрытии [4]. Пусть заданы множество $M = \{1, ..., k\}$ и набор его подмножеств $M_1, ..., M_n$ таких, что $\bigcup_{i=1}^{n} M_i = M$. Совокупность подмножеств $M_i, i \in N_0 \subseteq N = \{1, ..., n\}$, называется покрытием множества M, если $\bigcup_{i \in N_0} M_i = M$. Каждому M_i приписан единичный вес. Требуется найти покрытие минимального веса.

Сведем задачу о минимальном покрытии к задаче MP. Для этого рассмотрим частный случай задачи MP. Пусть множество предприятий $I = N$, а множество $J = J_1 \cup J_2$, где $|J_1| = |I|$ и $J_2 = M$. Определим бюджет каждого потребителя следующим образом: $b_j = 3, j \in J_1$, и $b_j = 2, j \in J_2$. Считаем, что $J_1 = \{j_i, i \in I\}$. Положим $c_{ij_i} = 0, j_i \in J_1$, и $c_{ij} = 0, j \in M_i$. Все остальные компоненты матрицы транспортных затрат положим равными 4, то есть в соответствии с ограничением (1.3) сделаем эти транспортные пути закрытыми. Заметим, что можно считать, что в оптимальном решении (p^*, x^*) задачи MP с выше указанными начальными данными компоненты p_i^* вектора цен принимают значения 2 или 3. Действительно, рассмотрим вектор цен $\tilde{p}$, компоненты которого определяются следующим образом. Если для некоторого $i \in I$ выполняется неравенство $p_i^* \leq 2$, то положим $\tilde{p}_i = 2$, и положим $\tilde{p}_i = 3$, если $p_i^* > 2$. Тогда решение $(\tilde{p}, x^*)$ будет допустимым, если выполняются огра-

ничения (1.2) и (1.3). Предположим, что хотя бы одно из ограничений (1.2) нарушено. Пусть $\sum_{i\in I}(b_j - c_{ij} - \tilde{p}_i)x^*_{ij} < 0$ для некоторого $j \in J$. Из допустимости решения (p^*, x^*) следует, что существует $i \in I$ такое, что $b_j - \tilde{p}_i < 0$, где b_j либо 2, либо 3. Откуда получаем, что $b_j - p^*_i < 0$ и (p^*, x^*) – недопустимое решение. Противоречие. Аналогично рассуждаем и в случае, когда нарушается ограничение (1.3). Следовательно, можно ограничиться рассмотрением допустимых решений задачи MP с дополнительным ограничением $p_i \in \{2, 3\}, i \in I$. В этом случае матрица x_{ij} определяется следующим образом: $x_{ij_i} = 1, j_i \in J_1$, и $x_{ij} = 1, j \in M_i$, если $p_i = 2$. Все остальные компоненты матрицы $x_{ij} = 0$. Таким образом, целевая функция имеет следующий вид:

$$f(p, x) = 2|\tilde{J}| + 2|\tilde{N}| + 3(|I| - |\tilde{N}|) \to \max_{\tilde{N}},$$

где $\tilde{N} \subset I$, а $\tilde{J} = \bigcup_{i\in\tilde{N}} M_i$. Покажем, что максимум достигается на $\tilde{N} = N^*_0$ - покрытии минимального веса. Допустим это не так. Тогда

$$(2|\tilde{J}| + 2|\tilde{N}| + 3(|I| - |\tilde{N}|)) - (2|J| + 2|N^*_0| + 3(|I| - |N^*_0|)) =$$

$$= (2|\tilde{J}| - |\tilde{N}|) - (2|J| - |N^*_0|) > 0.$$

Если $|\tilde{J}| = |J|$, тогда $\tilde{N}$ – покрытие, вес которого меньше веса оптимального покрытия – противоречие. Пусть $2(|J|-|\tilde{J}|) < |N^*_0|-|\tilde{N}|$ и $|\tilde{J}| < |J|$. Так как N^*_0 – покрытие, значит существует не более чем $|J| - |\tilde{J}|$ множеств M_i, где $i \in V \subset N^*_0 \setminus \tilde{N}, |V| \leq |J| - |\tilde{J}|$, для которых $\tilde{N} \cup V$ – покрытие. Так как $|V| \leq |J| - |\tilde{J}|$, то N^*_0 – не оптимальное покрытие.◁

Из теоремы 5 следует, что при условии $P \neq NP$ не существует алгоритма, находящего оптимальное решение задачи MP за полиномиальное или псевдополиномиальное время. Это означает тогда, что любой точный

алгоритм решения такой задачи будет иметь экспоненциальную трудоёмкость в худшем случае. В связи с этим возникает проблема разработки приближённых и точных алгоритмов решения, трудоёмкость которых достаточна мала, по крайней мере, для задач реальной размерности.

Пусть $a_{ij} := b_j - c_{ij}; i \in I, j \in J$. Тогда задачу MP можно переписать в виде следующей задачи квадратичного программирования MPR:

$$\sum_{i \in I} p_i \sum_{j \in J} x_{ij} \to \max_{p,x}$$

$$\sum_{i \in I} (a_{ij} - p_i) x_{ij} \geq 0; j \in J;$$

$$\sum_{i \in I} (a_{ij} - p_i) x_{ij} \geq a_{kj} - p_k; k \in I, j \in J;$$

$$\sum_{i \in I} x_{ij} \leq 1; j \in J;$$

$$p_i \geq 0, x_{ij} \in \{0, 1\}; i \in I, j \in J.$$

Без ограничения общности считаем, что для любого $j \in J$ существует $i \in I$, при котором $a_{ij} > 0$. Пусть $a_1 < ... < a_\omega$ – строго упорядоченная по возрастанию последовательность всех $a_{w(j)} := \max_{i \in I} a_{ij}, j \in J$. Положим $S_k := \{j \in J : a_k = \max_{i \in I} a_{ij}\}$, $s_k := |S_k|$. Обозначим через g^* сумму $\sum_{1 \leq k \leq \omega} a_k s_k$, которая есть в точности сумма бюджетов всех потребителей за вычетом их минимально возможных транспортных затрат. Данная величина является оптимальным значением целевой функции задачи DP. Очевидно, что g^* – верхняя граница задачи MPR. Пусть $g_k := \sum_{0 \leqslant t \leqslant k} a_{\omega - t} s_{\omega - t}$. Под матрицами $(a_{ij}^k), 1 \leqslant k \leqslant \omega$, будем понимать матрицы размера $m \times n$ с элементами $a_{ij}^k := a_{ij}$, если $a_{ij} \geqslant a_{\omega - k}$, и 0 в противном случае. За MPR_k обозначим задачу MPR с матрицей (a_{ij}^k) и равномерным ценообразованием (т.е. задача UP с матрицей (a_{ij}^k)).

Другими словами, к задаче MPR с матрицей $(a_{ij}) := (a^k_{ij})$ добавляется ограничение $p_\alpha = p_\beta, \forall \alpha, \beta \in I$. Пусть $Opt(k)$ – оптимальное значение целевой функции задачи MPR_k. Докажем следующее утверждение.

Лемма 2 $Opt(k) = a_{\omega-k+e}(s_\omega + ... + s_{\omega-k+e})$, $0 \leqslant e \leqslant k$.

Доказательство ▷ В теореме 1 для задачи MP было показано, что в оптимальном решении существует хотя бы одно предприятие из I_s. Не трудно показать, что этот факт верен и для задачи UP, воспользовавшись идеей доказательства теоремы 1. Тогда для оптимального решения (p^*, x^*) задачи MPR_k существуют такие $i \in I, j \in J$, что $x^*_{ij} = 1$ и $a_{ij} - p^* = 0$. И, согласно ограничению (1.3), для задачи MPR_k оптимальная цена $p^* = \max_{i \in I} a_{ij} = a_{\omega-k+e}$, для некоторого $0 \leqslant e \leqslant k$. А из ограничений (1.2) и (1.3) следует, что потребитель j обслуживается и приносит доход p^*, если соответствующая этому потребителю величина $a_{w(j)}$ не меньше этой цены. Получаем $Opt(k) = a_{\omega-k+e}(s_\omega + ... + s_{\omega-k+e})$ для некоторого $0 \leqslant e \leqslant k$. ◁

Следствие 1 *Функция $Opt(k)$ монотонно неубывающая.*

Доказательство ▷ Утверждение следует из равенства

$$Opt(k+1) = \max\{Opt(k), a_{\omega-k-1}(s_\omega + ... + s_{\omega-k-1})\},$$

которое следует из леммы 2.◁

Следствие 2 *Оптимальные значения целевых функций задач MPR_k и MPR_{k+1}, $1 \leqslant k \leqslant \omega - 1$, совпадают тогда и только тогда, когда: $Opt(k) \geq a_{\omega-k-1}(s_\omega + ... + s_{\omega-k-1})$.*

Пусть $H_k := 1 + 1/2 + 1/3 + ... + 1/(\sum\limits_{0 \leqslant t \leqslant k} s_{\omega-t})$. Докажем следующую лемму.

Лемма 3 *Для любого k выполняется неравенство $g_k/Opt(k) \leqslant H_k$.*

Доказательство ▷ Докажем утверждение леммы индукцией по k.

$k = 0$: Очевидно, что $Opt(0) = a_\omega s_\omega = g_0$.

Пусть для $k = z$ выполнено $g_z/Opt(z) \leqslant H_z$. И допустим, что для $k = z + 1$ это неравенство не выполнено, то есть $g_{z+1}/Opt(z+1) > H_{z+1}$. Рассмотрим два случая.

Случай 1. Пусть оптимальные значения целевых функций задач MPR_z и MPR_{z+1} совпадают. Рассмотрим отношение g_{z+1} к $Opt(z+1)$:

$$\frac{g_{z+1}}{Opt(z+1)} = \frac{g_z + a_{\omega-z-1}s_{\omega-z-1}}{Opt(z)} = \frac{g_z}{Opt(z)} + \\ + \frac{a_{\omega-z-1}s_{\omega-z-1}}{Opt(z)} = \frac{g_z}{Opt(z)} + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z+e}(s_\omega + ... + s_{\omega-z+e})}.$$

Первое равенство следует из определения величин g_{z+1}, g_z и совпадения величин $Opt(z)$ и $Opt(z+1)$, второе равенство очевидно, а третье следует из леммы 2. Итак, получили, что:

$$\frac{g_{z+1}}{Opt(z+1)} = \frac{g_z}{Opt(z)} + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z+e}(s_\omega + ... + s_{\omega-z+e})}.$$

Таким образом, исходное неравенство $g_{z+1}/Opt(z+1) > H_{z+1}$ можно переписать в следующем виде:

$$H_{z+1} < \frac{g_z}{Opt(z)} + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z+e}(s_\omega + ... + s_{\omega-z+e})} \leqslant H_z + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z+e}(s_\omega + ... + s_{\omega-z+e})} \leqslant \\ \leqslant H_z + \frac{a_{\omega-z-1}s_{\omega-z-1}}{(a_{\omega-z-1}(s_\omega + ... + s_{\omega-z-1})} = H_z + \frac{s_{\omega-z-1}}{(s_\omega + ... + s_{\omega-z-1})}.$$

Здесь мы воспользовались индукционным переходом и следствием 2, согласно которому $a_{\omega-z+e}(s_\omega + ... + s_{\omega-z+e}) \geqslant a_{\omega-z-1}(s_\omega + ... + s_{\omega-z-1})$, для

некоторого $0 \leqslant e \leqslant z$. Эти преобразования приводят нас к следующему результату:

$$H_{z+1} < H_z + \frac{s_{\omega-z-1}}{s_\omega + ... + s_{\omega-z-1}}.$$

Теперь, воспользовавшись определением величины H_z, получаем следующее неравенство:

$$\frac{1}{s_\omega + ... + s_{\omega-z} + 1} + ... + \frac{1}{s_\omega + ... + s_{\omega-z-1}} < \frac{s_{\omega-z-1}}{s_\omega + ... + s_{\omega-z-1}}.$$

В левой части последнего неравенства ровно $s_{\omega-z-1}$ слагаемых со знаменателями не превосходящими сумму $s_\omega + ... + s_{\omega-z-1}$, а значит левая часть не меньше правой — противоречие.

Случай 2. Пусть оптимальные значения целевых функций задач MPR_z и MPR_{z+1} не совпадают. Тогда из следствия 2 следует, что выполняется неравенство $Opt(z) < a_{\omega-z-1}(s_\omega + ... + s_{\omega-z-1})$. А из доказательства следствия 1 следует, что для любого $0 \leqslant e \leqslant z$ выполняется $a_{\omega-z+e}(s_\omega+...+s_{\omega-z+e}) < a_{\omega-z-1}(s_\omega+...+s_{\omega-z-1})$. Рассмотрим отношение g_{z+1} к $Opt(z+1)$:

$$\frac{g_{z+1}}{Opt(z+1)} = \frac{g_z}{Opt(z+1)} + \frac{a_{\omega-z-1}s_{\omega-z-1}}{Opt(z+1)} = \frac{g_z}{Opt(z+1)} + \\ + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z-1}(s_\omega + ... + s_{\omega-z-1})} < \frac{g_z}{Opt(z)} + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z-1}(s_\omega + ... + s_{\omega-z-1})}.$$

Первые два равенства очевидны. Неравенство следует из того, что по условию $Opt(z+1) > Opt(z)$. Получаем что:

$$\frac{g_{z+1}}{Opt(z+1)} < \frac{g_z}{Opt(z)} + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z-1}(s_\omega + ... + s_{\omega-z-1})}.$$

Следовательно, исходное неравенство $g_{z+1}/Opt(z+1) > H_{z+1}$ можно переписать в следующем виде:

$$H_{z+1} < \frac{g_z}{Opt(z)} + \frac{a_{\omega-z-1}s_{\omega-z-1}}{a_{\omega-z-1}(s_\omega + ... + s_{\omega-z-1})} \leqslant H_z + \frac{s_{\omega-z-1}}{(s_\omega + ... + s_{\omega-z-1})}.$$

Второе неравенство следует из индукционного предположения. Эти преобразования приводят нас к следующему результату:

$$H_{z+1} < H_z + \frac{s_{\omega-z-1}}{s_\omega + ... + s_{\omega-z-1}},$$

который, как показано в первом случае, приводит к противоречию. Следовательно, индукционный переход доказан, откуда и следует утверждение леммы. ◁

Теорема 6 *Задача MP принадлежит классу Log-APX.*

Доказательство ▷ Оптимизационная задача принадлежит к классу Log-APX, тогда и только тогда, когда существует полиномиальный алгоритм, находящий $(c\ln(r))$-приближенное решение задачи, где r — длина входа, а c — некоторая константа [27]. Пусть Opt_{MPR} – оптимальное значение задачи MPR (а значит и MP). Из определения задачи MPR_w следует, что любое его оптимальное решение является допустимым решением задачи MPR (а значит и MP) со значением целевой функции равным $Opt(w)$. Тогда из леммы 3 мы имеем следующее неравенство $Opt_{MPR}/Opt(w) \leqslant g_w/Opt(w) \leqslant H_w$, где $Opt(w)$ есть оптимальное значение задачи MPR_w с одинаковой ценой на всех предприятиях, а $g_w = g^*$ – верхняя граница для Opt_{MPR}. Очевидно, что величина $Opt(w)$, оптимальная цена на всех предприятиях и, соответствующие, переменные x_{ij} вычисляются за полиномиальное время. Теперь утверждение теоремы следует из того, что значение H_w с помощью неравенства Стирлинга оценивается величиной порядка $c\ln(m)$, где $m = |J|$. ◁

Теорема 7 *При условии P≠NP для задачи MP не существует полиномиального приближенного алгоритма с абсолютной погрешностью,*

ограниченной константой.

Доказательство ▷ Допустим обратное. Пусть для некоторого полиномиального алгоритма A верна оценка: $f(p^*, x^*) - f(p^A, x^A) < W$, где f – целевая функция, а $W = const > 0$. Без ограничения общности можно считать, что W – целое число. Покажем, что с помощью алгоритма A можно построить полиномиальный алгоритм для решения задачи MP, которая является NP-трудной.

Так как за полиномиальное время можно построить оптимальное решение задачи MP при фиксированных переменных x_{ij}, то будем считать, что алгоритм A устроен так, что переменные p_i^A оптимальны для заданных переменных x_{ij}^A. Пусть L произвольный вход задачи MP. Определим

$$Opt(L) = f(p^*(L), x^*(L)) \text{ и } A(L) = f(p^A(L), x^A(L)).$$

Пусть исходные данные индивидуальной задачи L' получены из исходных данных задачи L умножением векторов (b_j) и (c_{ij}) на величину W. Без ограничения общности считаем, что (b_j) и (c_{ij}) являются целыми числами. Заметим, что теорема 1, и как следствие лемма 1 и теорема 2, верны для задачи MP с фиксированной матрицей назначений. Можно считать, что $p^* \in Price$ и алгоритм A находит решение (p^A, x^A), где $p^A \in Price$. Тогда из определения множества $Price$ следует, что величины $p^*(L)$, $p^A(L)$, $p^*(L')$, $p^A(L')$ - целые числа, причем $p^*(L')$ и $p^A(L')$ кратны W, а также верно соотношение: $Opt(L') = W \cdot Opt(L)$. Следовательно

$$Opt(L') - A(L') = W \cdot Opt(L) - A(L') < W.$$

Откуда получаем

$$Opt(L) - A(L')/W = f(p^*(L), x^*(L)) - f(p^A(L')/W, x^A(L')) < 1,$$

где $f(p^*(L), x^*(L))$ и $f(p^A(L')/W, x^A(L'))$ - целые числа. Тогда

$$f(p^*(L), x^*(L)) = f(p^A(L')/W, x^A(L')),$$

и решение $(p^A(L')/W, x^A(L'))$ - оптимальное решение задачи MP на входе L. Таким образом задача MP является полиномиально разрешимой, что противоречит условию P≠NP. ◁

2.3 Гибридные алгоритмы для задачи фабричного ценообразования

В данном разделе для решения задачи фабричного ценообразования предлагается базовый гибридный точный алгоритм, использующий метаэвристики [79] и декомпозицию [53]. Этот алгоритм стал основой ряда приближенных и точных алгоритмов, вычислительное исследование которых проводилось в рамках данной работы. Интерес к подобного рода итерационным алгоритмам связан с их полезным свойством порождать верхние и нижние оценки на значение искомого оптимума.

Сформулируем в общем виде метод декомпозиции применительно к максиминным задачам. Аналогичные построения и рассуждения верны и для минимаксных задач. С этой точки зрения декомпозицию Бендерса можно рассматривать как сведение исследуемой задачи к эквивалентной максиминной задаче (координирующая задача или master problem в классической формулировке [53]).

Рассмотрим следующую максиминную задачу:

$$z = \max_{x \in X} \min_{y \in Y} R(x, y).$$

В дальнейшем будем предполагать, что хотя бы одно из множеств X или Y конечно и задача разрешима. Перепишем её в эквивалентном виде, который обычно используется при описании декомпозиции Бендерса:

$$\gamma \to \max_{\gamma, x \in X}$$

$$\gamma \leq R(x, y), y \in Y.$$

Сформулируем метод декомпозиции для максиминной задачи. Пусть $\overline{Y} \subseteq Y$.

Шаг 1 Решить релаксированную задачу

$$\overline{z} = \max_{\gamma, x \in X} \gamma$$

$$\gamma \leq R(x, y), y \in \overline{Y}$$

Пусть x^* – её оптимальное решение.

Шаг 2: Решить подзадачу

$$\underline{z} = \min_{y \in Y} R(x^*, y)$$

Пусть y^* – её оптимальное решение.

Шаг 3: Если $\overline{z} = \underline{z}$, то стоп. Иначе $\overline{Y} := \overline{Y} \cup y^*$ и вернуться на **шаг 1**.

Если на шагах 1 и 2 существуют оптимальные решения x^* и y^*, то $\underline{z} \leq z \leq \overline{z}$. При этом, генерируемая методом последовательность $\overline{z}$ является невозрастающей. Если на шаге 3 выполняется условие $\overline{z} = \underline{z}$, то $z = \overline{z} = \underline{z}$ – искомое оптимальное значение максиминной задачи, а

(x^*, y^*) – её оптимальное решение. Для упрощения анализа этого метода будем предполагать, что разрешима любая релаксированная задача, возникающая на шаге 1. Аналогично, будем считать, что разрешима любая подзадача на шаге 2. Тогда очевидно, что описываемый метод конечен, если на шагах 1 или 2 повторяются оптимальные решения. По-видимому, впервые это было отмечено в [53]. Там же доказано, что если хотя бы одно из множеств X или Y конечно, то конечен и метод. В этом случае также можно отказаться от требования, что всегда разрешимы задачи возникающие на шагах 1 и 2. Чтобы гарантировать корректную работу алгоритма будем считать, что в случае неразрешимости задачи на шаге 1 генерируется вектор $x^* \in X$, который не встречался на предыдущих итерациях. Аналогично поступаем, если неразрешима подзадача на шаге 2.

Важным при реализации этого метода является выбор начального семейства решений $\overline{Y}$. Из описания метода следует, если множество X конечно, то вне зависимости от того конечно или бесконечно множество Y основная доля ограничений, генерируемая с помощью решений, является несущественной. Поэтому возникает проблема отыскания таких начальных множеств $\overline{Y}$, которые позволяют быстро найти оптимальное решение исследуемой задачи. Именно здесь открывается простор для использования метаэвристик.

Идея использовать декомпозицию для решения оптимизационных задач восходит к работам Данцига, Вольфа и Бендерса [53, 69, 81]. Существенный прогресс был достигнут при разработке декомпозиционных алгоритмов для задач размещения производства [69]. В области двух-

уровневых комбинаторных задач размещения первый точный алгоритм декомпозиционного типа предложен в [23] для решения двухуровневой задачи о конкурентной p-медиане (задача о (r|p)-центроиде). Данный алгоритм и его модификации в настоящее время наиболее эффективные точные алгоритмы решения указанной задачи. Идея модифицированных алгоритмов в более глубокой интеграции локального поиска, метаэвристик и декомпозиции.

В [14] для задачи ценообразования предложен гибридный точный алгоритм. Однако этот алгоритм оказался не очень удобным в реализации, т.к. в релаксированной координирующей задаче целевая функция получилась кусочно линейной, что существенно ограничило применимость библиотек типа CPLEX при практической реализации декомпозиции. В данном разделе предлагается алгоритм, в котором релаксированная координирующая задача является линейной задачей булевого программирования с одной непрерывной переменной, а подзадача, которая также решается на каждой итерации алгоритма, является задачей линейного программирования, что существенно упрощает программную реализацию и вычислительный эксперимент.

Перепишем модель первого раздела в виде линейной задачи частично-целочисленного программирования. Обозначим за $\overline{p}_i = \max_j(b_j - c_{ij})$ – максимально возможную цену на i-м предприятии, и введем переменную $z_{ij} \geq 0$ – доход, который получает производитель на i-м предприятии от j-го потребителя. Используя данные обозначения, получим:

$$\max_{p,x,z} \sum_{i \in I} \sum_{j \in J} z_{ij}; \tag{2.1}$$

$$\sum_{i\in I}(b_j - c_{ij})x_{ij} - \sum_{i\in I} z_{ij} \geq 0, \quad j \in J; \tag{2.2}$$

$$c_{kj} + p_k - \sum_{i\in I} c_{ij}x_{ij} - \sum_{i\in I} z_{ij} \geq 0, \quad k \in I,\ j \in J; \tag{2.3}$$

$$(1 - x_{ij})\overline{p}_i - z_{ij} + p_i \geq 0, \quad i \in I,\ j \in J; \tag{2.4}$$

$$(1 - x_{ij})\overline{p}_i + z_{ij} - p_i \geq 0, \quad i \in I,\ j \in J; \tag{2.5}$$

$$z_{ij} \leq \overline{p}_i x_{ij}, \quad i \in I,\ j \in J; \tag{2.6}$$

$$\sum_{i\in I} x_{ij} \leq 1, \quad j \in J; \tag{2.7}$$

$$p_i \geq 0, x_{ij} \in \{0, 1\}, z_{ij} \geq 0, \quad i \in I,\ j \in J. \tag{2.8}$$

Ограничения (2.4) – (2.6) гарантируют, что доход производителя z_{ij} от обслуживания j-го потребителя на i-м предприятии равен p_i, если j-й потребитель выбрал i-е предприятие, и 0 в противном случае.

Применим к задаче подход, реализованный в [14], и получим точный гибридный декомпозиционный алгоритм решения. Задача (2.1) – (2.8) при фиксированном x является задачей линейного программирования. Введем двойственные переменные $t \geq 0$, $\mu \geq 0$, $\lambda_1 \geq 0$, $\lambda_2 \geq 0$, $\lambda_3 \geq 0$, для ограничений (2.2) – (2.6). Обозначим через $\delta(x,\ t, \mu,\ \lambda_1, \lambda_2, \lambda_3)$ целевую функцию задачи $D(x)$ двойственной к задаче (2.1) – (2.8) при фиксированном x, а через $\delta^1_{ij}(x,\ t, \mu,\ \lambda_1, \lambda_2, \lambda_3)$, $\delta^2_i(x,\ t, \mu,\ \lambda_1, \lambda_2, \lambda_3)$ ограничения задачи $D(x)$ соответствующие прямым переменным z_{ij} и p_i. Так как задача (2.1) – (2.8) разрешима, то задача $D(x)$ при любом x имеет допустимые решения. И, следовательно, она может быть неразрешимой только при наличии в допустимой области бесконечного ребра, вдоль которого неограниченно убывает целевая функция [3]. В дальнейшем этот факт будет использован в декомпозиционном алгоритме при порождении

отсечений, названных в соответствии со сложившейся традицией отсечениями допустимости. В соответствии с этой традицией, отсечения, порождаемые с помощью оптимальных вершин, полученных при решении задачи $D(x)$, называются отсечениями оптимальности.

Приведём описание точного гибридного алгоритма для решения задачи (2.1) – (2.8).

Шаг 1 Случайным образом (либо с помощью алгоритма генетического локального поиска для задачи (2.1) – (2.8) [13]) построить семейство решений $x^r, r = \overline{1, R}$, для каждого из которых решим следующую задачу:

$$\rho(x^r) = \min_{t,\mu,\lambda_1,\lambda_2,\lambda_3} \delta(x^r, t, \mu, \lambda_1, \lambda_2, \lambda_3)$$
$$\delta^1_{ij}(x^r, t, \mu, \lambda_1, \lambda_2, \lambda_3) \le 0,\ \ i \in I, j \in J;$$
$$\delta^2_i(x^r, t, \mu, \lambda_1, \lambda_2, \lambda_3) \le 0,\ \ i \in I.$$

Найдем оптимальные значения двойственных переменных $(t^r, \mu^r, \lambda^r_1, \lambda^r_2, \lambda^r_3)$, если задача разрешима. Иначе, в качестве вектора $(t^r, \mu^r, \lambda^r_1, \lambda^r_2, \lambda^r_3)$ возьмем направляющий вектор соответствующего бесконечного ребра. Предположим, что задача $D(x^r)$ разрешима для всех $r = 1, \ldots, Q$. Положим $\mathbf{LB} := \max\{\delta(x^r, t^r, \mu^r, \lambda^r_1, \lambda^r_2, \lambda^r_3), r = \overline{1, Q}\}$, $U = R - Q$;

Шаг 2 Решить релаксированную координирующую задачу:

$$\max_{x_{ij}\in\{0,1\}, y\ge 0} y$$
$$y \le \delta(x, t^q, \mu^q, \lambda^q_1, \lambda^q_2, \lambda^q_3), q = \overline{1, Q}, \text{ (отсечение оптимальности);}$$
$$\delta(x, t^u, \mu^u, \lambda^u_1, \lambda^u_2, \lambda^u_3) \ge 0, u = \overline{1, U}, \text{ (отсечение допустимости);}$$
$$1 - \sum_{i\in I} x_{ij} \ge 0,\ \ j \in J.$$

Пусть $(\overline{y}, \overline{x})$ – оптимальное решение, $\mathbf{UB} := \overline{\mathbf{y}}$ – верхняя граница.

Шаг 3: Решить подзадачу $\rho(x)$ с $x = \overline{x}$.

1) Задача разрешима: Если $\mathbf{UB} = \rho(\overline{\mathbf{x}})$, то стоп. Нашли оптимальное

решение исходной задачи. Иначе положить: $Q := Q + 1, t^Q := t, \mu^Q := \mu, \lambda_1^Q := \lambda_1, \lambda_2^Q := \lambda_2, \lambda_3^Q := \lambda_3$, где t, μ и $\lambda_1, \lambda_2, \lambda_3$ – оптимальные двойственные переменные. Если $\rho(\overline{\mathbf{x}}) > \mathbf{LB}$, то $\mathbf{LB} := \rho(\overline{\mathbf{x}})$ – новая нижняя граница. Далее на шаг 2.

2) Задача не разрешима: Найдем направляющий вектор бесконечного ребра $(t, \mu, \lambda_1, \lambda_2, \lambda_3)$ и положим: $U := U + 1, t^U := t, \mu^U := \mu, \lambda_1^U := \lambda_1, \lambda_2^U := \lambda_2, \lambda_3^U := \lambda_3$. Далее на **шаг 2**.

Данный алгоритм решения задачи конечен [14]. Если на шаге 3 заменить условие совпадения верхней и нижней границ на неравенство $\frac{UB-LB}{UB} < \varepsilon$, где $\varepsilon > 0$, то получим приближённую версию алгоритма.

2.4 Вычислительный эксперимент

Из описания базового алгоритма видно, что если отбросить шаг 1, то получится классический алгоритм Бендерса [82]. Основной недостаток таких алгоритмов – медленная сходимость. Поэтому в работе значительное внимание уделялось поиску методов ускорения декомпозиционных алгоритмов. В настоящее время имеется большое количество исследований на эту тему, в которых разрабатываются быстрые модификации декомпозиционных алгоритмов [69, 82]. В [14, 22] эта проблема рассматривалась с точки зрения применения метаэвристик.

Так как релаксированная координирующая задача, возникающая на шаге 2, является задачей смешанного программирования с булевыми переменными, то для ее решения на каждой итерации базового алгоритма требуются значительные вычислительные ресурсы. Поэтому основная доля исследований на эту тему посвящена или уменьшению числа

решаемых на шаге 2 релаксированных координирующих задач, или различным процедурам, упрощающим нахождение решения текущей релаксированной координирующей задачи. В [68] было впервые предложено порождать отсечения, используя решения линейной релаксации релаксированной координирующей задачи. Эта идея основана на том, что любая вершина (или любой крайний луч [3]) задачи двойственной к подзадаче порождает отсечение оптимальности (или допустимости) для релаксированной координирующей задачи. Таким образом, вместо сложной задачи на шаге 2 решается задача линейного программирования. Другой подход был предложен в [54]. Вместо того, чтобы искать оптимальное решение релаксированной координирующей задачи, авторы предложили остановить процесс решения на некотором допустимом решении, которое затем используется на шаге 3 для порождения отсечения оптимальности (или допустимости). Основной минус таких упрощений в том, что нельзя гарантировать сходимость процесса. И проще всего использовать такие идеи для генерации приближённых решений. Либо начиная с какого-то момента, накопив достаточно большое множество отсечений с небольшими затратами вычислительных ресурсов, вернуть процесс к исходной схеме и таким образом гарантировать сходимость. Далее для задачи фабричного ценообразования предлагаются два подхода – двухфазная схема и двухфазная схема с контролируемым числом ограничений. Эти схемы используются для ускорения базового алгоритма.

Перечислим все тестируемые алгоритмические схемы, после чего перейдем к их непосредственному описанию:

1. базовая схема (шаги 1-3) (**General scheme**);

2. двухфазная схема (**2-round scheme**);

3. двухфазная схема с приближенной версией алгоритма на 1-й фазе (**App. 2-r.**);

4. двухфазная схема с контролируемым числом ограничений (**Box**);

5. упрощенная двухфазная схема (**Simplified 2-round scheme**).

Первые четыре схемы являются точными, а схема 5 дает неплохую верхнюю оценку. Схемы 2 – 4 позволяют уменьшить число итераций и время счета базовой схемы.

Идея двухфазного метода (двухфазной схемы) основывается на хорошо известном факте из теории. Если известно множество отсечений, которое определяет выпуклую оболочку множества допустимых решений целочисленной задачи, то декомпозиционный алгоритм находит оптимальное целочисленное решение за одну итерацию [69]. Множество отсечений, определяющих допустимую область линейной релаксации целочисленной задачи с вычислительной точки зрения найти легче и оно содержит выпуклую оболочку допустимых решений целочисленной задачи. Поэтому в алгоритмах основанных на двухфазной схеме сначала решается линейная релаксация задачи (2.1) – (2.8) приведенным выше базовым алгоритмом, у которого на шаге 2 решается релаксированная координирующая задача с непрерывными переменными x_{ij}. Затем при решении исходной задачи (2.1) – (2.8) к семейству отсечений шага 1 добавляется оптимальное семейство отсечений, полученное при решении линейной задачи.

Идея схемы 3 заключается в том, чтобы ограничить время работы базового алгоритма решения задачи (2.1) – (2.8) на 1-ой фазе, когда ре-

шается задача с непрерывными переменными x_{ij}. Для этого выбирается $0 < \varepsilon < 1$, которое используется для ограничения времени выполнения 1-й фазы. При выполнении неравенства $\frac{UB-LB}{UB} < \varepsilon$ осуществлялся переход ко 2-й фазе.

Идея схемы 4 заключается в том, чтобы контролировать число отсечений. Эта схема является гибридом двухфазной схемы и идей поиска с запретами [74]. На каждой фазе используется три списка. Два ограниченных, а третий произвольной длины. Один из ограниченных списков предназначен для хранения отсечений оптимальности, а другой для хранения отсечений допустимости. Когда добавляется новое отсечение оптимальности, то из соответствующего списка выбрасывается самое старое отсечение. Если верхняя граница, определяемая текущей релаксированной координирующей задачей оказывается меньше предыдущей, то выброшенное ограничение возвращается. Оно переводится в неограниченный список. Аналогичным образом выполняются операции при работе с отсечениями допустимости.

В упрощённой двухфазной схеме вычисления завершаются на шаге 2 на первой итерации второй фазы.

Все эти схемы тестировались на примерах небольших размерностей до 30 предприятий и 50 потребителей, в которых значения величин b_j и c_{ij} выбирались равновероятно из интервала [1, 99]. Тесты проводились на 20-ти примерах каждой размерности. Чтобы не ограничивать размерность решаемых задач, время работы 1 – 4 схем ограничивалось получасом (см. таблицы 2.1, 2.2 и 2.3), а упрощённой схемы – часом (см. таблицу 2.4). Так как время работы и точность (в том случае, когда время работы

алгоритма ограничивалось) базовой схемы оказались на порядок хуже остальных, и зачастую этот метод даже не находил допустимое решение за ограниченное время, то результат тестирования первой схемы в ниже приведенных таблицах отсутствует. Стоит отметить, что время работы и точность схем 2 – 4, как показал эксперимент, в среднем отличаются друг от друга не более чем в 2-3 раза.

В таблицах 2.1, 2.2 и 2.3 представлены результаты экспериментов, которые проводились для сравнения трудоёмкости точных схем. В таблицах 2.1 и 2.2 представлены относительная погрешность (величина $\frac{UB-LB}{UB}$) и время работы схемы 2 и схем 3 при выборе $\varepsilon = 0.02, 0.05, 0.1, 0.2$. В таблице 2.3 – относительная погрешность и время работы схемы 4, при этом эксперимент проводился при суммарном количестве отсечений оптимальности и допустимости равном 20, 30 и 60.

Эксперимент со схемами 2 и 3 (см. таблицы 2.1 и 2.2) показал, что на небольших размерностях $(5 \times 10 - 20)$ для ускорения счета можно часть отсечений линейной задачи выбросить. Из таблиц 2.1 и 2.2 видно, что схема 2 проигрывает по времени схеме 3 при выборе $\varepsilon = 0.02, 0.05$, когда число потребителей 10 и 20, и $\varepsilon = 0.02$, когда число потребителей 15. При переходе к большим размерностям с ростом числа итераций появляется необходимость переносить во вторую фазу все отсечения линейной задачи. Здесь существенной проблемой становиться плохая сходимость декомпозиционных алгоритмов. В среднем, число отсечений, генерируемых для получения оптимального значения, практически не отличается для схем 2 и 3. Поэтому, убирая часть отсечений линейной задачи в схеме 3, приходится дополнять их гораздо более дорогостоящими отсечениями

Dimen.		2-round		App. 2-r.(0.02)		App. 2-r.(0.05)	
n	m	deviat.	time	deviat.	time	deviat.	time
5	10	0	8.8"	0	8.5"	0	8.4"
5	15	0	90"	0	69"	0	110"
5	20	0.013	956"	**0.009**	776"	0.017	848"
5	25	**0.076**	1544"	0.104	1821"	0.135	1831"
5	30	**0.166**	1823"	0.179	1821"	0.207	1825"

Таблица 2.1: Результат работы схем 2 и 3

Dimen.		App. 2-r.(0.1)		App. 2-r.(0.2)	
n	m	deviat.	time	deviat.	time
5	10	0	12.1"	0	17.7"
5	15	0	133"	$<$0.001	315"
5	20	0.033	1094"	0.057	1293"
5	25	0.183	1807"	0.224	1815"
5	30	0.244	1822"	0.327	1822"

Таблица 2.2: Результат работы схем 2 и 3 (продолжение)

булевой задачи, что приводит к увеличению времени работы алгоритмов на основе данной схемы с ростом ε. Схема 4 при сравнении со схемами 2 и 3 оказалась вполне конкурентоспособной. Из таблицы 2.3 видно, что схема Box при выборе разного суммарного количества отсечений оптимальности и допустимости имеет практически одинаковые показатели времени счета и точности, которые не уступают показателям схемы 3 с $\varepsilon = 0.2$ на размерностях $5 \times 10 - 15$, а на размерностях $5 \times 20 - 25$ с $\varepsilon = 0.1, 0.2$, и проигрывает только схеме 3 с $\varepsilon = 0.02$ на размерности 5×30. Но стоит отметить, что на больших размерностях схема 4 уступает совсем немного схеме 2 (меньше чем в 1.3 раза по точности). Эксперимент с двухфазной схемой показал также, что с ростом размерности задачи отсечения её линейной релаксации всё хуже и хуже аппроксимируют выпуклую

Dimen.		Box(10)		Box(15)		Box(30)	
n	m	deviat.	time	deviat.	time	deviat.	time
5	10	0	17.5"	0	21"	0	15.9"
5	15	0	192"	0	205"	0	232"
5	20	0.025	1251"	0.025	1343"	0.031	1348"
5	25	0.133	1748"	0.129	1724"	0.139	1785"
5	30	0.187	1705"	0.197	1709"	0.199	1712"

Таблица 2.3: Результат работы схемы 4

оболочку допустимых решений, что выражалось в стремительном росте числа итераций 2-ой фазы: 1 итерация в среднем для размерности 5×5, на порядок больше для размерности 5×10 и на 2 порядка больше для размерности 5×15 и т.д. Тем не менее, казалось бы естественная попытка в этой ситуации уменьшить количество отсечений генерируемых на первой фазе нередко приводила к росту числа итераций второй фазы и общего времени счета (схема 3). Таким образом с точки зрения уменьшения общей трудоёмкости вычислений очень важно качество аппроксимации на первой фазе. Как показывает следующий эксперимент с приближённой схемой это позволяет гарантировать и хорошие верхние оценки. В упрощённой двухфазной схеме вычисления завершаются на шаге 2 одной из первых итераций второй фазы, что позволяет получить достаточно неплохую верхнюю границу задачи фабричного ценообразования. Чтобы упростить анализ результатов в эксперименте вычисления завершались на шаге 2 первой итерации. Также ограничивалось время работы первой фазы одним часом. Для некоторых примеров размерностей 20×40 и 30×30 за отведенный час не удалось оптимально решить задачу, возникающую на первой фазе. При меньших размерностях тако-

Dimen.		Simplified 2-round		Linear	
n	m	deviat.	time	deviat.	time
5	10	0.0458	7.25"	0.125	<1"
5	15	0.0627	11.65"	0.142	<1"
5	30	0.136	199.5"	0.215	1.15"
5	50	0.192	1722.6"	0.273	4.75"
10	20	0.115	171.15"	0.179	1,65"
10	30	0.15	1032.45"	0.212	4.5"
20	20	0.111	1080.8"	0.165	5.8"
20	40	0.26	4025.6"	0.236	156.8"
30	30	0.204	3292.05"	0.201	202.1"

Таблица 2.4: Результат работы схемы 5

го не наблюдалось. В 3-й колонке таблицы 2.4 записана величина $\frac{UB-LB}{UB}$, а в 4-й – время работы упрощенной двухфазной схемы. В последних двух колонках величина $\frac{Lin-Opt}{Lin}$, где Lin – оптимальное значение задачи (2.1) – (2.8) с непрерывными переменными x_{ij}, и время работы пакета CPLEX на этой задаче.

Из таблицы 2.4 видно, что относительное уклонение верхней границы превысило относительное уклонение оптимального решения задачи (2.1) – (2.8) с непрерывними переменными x_{ij} только при размерностях 20×40 и 30×30. И здесь можно улучшить результаты, если останавливаться не на первой итерации второй фазы, а просто ограничить время счета.

2.5 Основные результаты второй главы

В этой главе рассмотрены задача ценообразования с фабричной, равномерной и дискриминационной стратегиями ценообразования. Показано, что задача фабричного ценообразования является NP-трудной в сильном

смысле, а задачи равномерного и дискриминационного ценообразования полиномиально разрешимы. Предложены два полиномиально разрешимых случая первой задачи, а также показано, что она лежит в классе $Log\text{-}APX$, и что для нее не существует полиномиального приближенного алгоритма с абсолютной погрешностью ограниченной константой при условии $P \neq NP$.

Для задачи со стратегией фабричного ценообразования разработан и исследован гибридный алгоритм (в том числе его модификации), основанный на идеи декомпозиции. Вычислительный эксперимент показал, что двухфазная схема данного алгоритма и двухфазная схема с контролем ограничений позволяют получить алгоритмы декомпозиционного типа более быстрые, чем другие. В том числе, эксперимент с упрощённой двухфазной схемой показал, что с помощью алгоритмов такого типа можно получить достаточно неплохие верхние оценки. Однако, этого недостаточно для получения быстрых точных и приближённых алгоритмов гибридного типа.

Глава 3

Задачи размещения производства и ценообразования

В этой главе показано, что рассматриваемые задачи размещения и ценообразования являются NP-трудными в сильном смысле, а также допускают построение эффективного приближенного алгоритма с полиномиальной относительной погрешностью, что является оценкой сверху на положение этих задач в аппроксимационной иерархии. Для задач с I типом размещения предприятий, когда за открытие предприятия взимается плата, показано, что они являются полными в классе $Poly\text{-}APX$ относительно AP-сводимости, тем самым получена еще и оценка снизу на положение данных задач в аппроксимационной иерархии.

Как и для задачи фабричного ценообразования в этой главе внимание сосредоточено на построении для задачи размещения (I) и фабричного ценообразования приближенного алгоритма, основанного на эвристических подходах. А именно гибридизация генетического алгоритма, поиска с чередующимися окрестностями и локального поиска.

3.1 Вычислительная сложность

При исследовании сложности нахождения оптимального или даже допустимого решения оптимизационной задачи одним из ключевых вопросов является связь данной задачи с полиномиальной иерархией задач распознавания. Обычно ограничиваются рассмотрением только нулевого и первого уровня иерархии, а именно классов P, NP и *co*-NP. Касательно исследуемых в этой статье задач имеет место следующее утверждение:

Теорема 8 *Задачи LDP, LMP, LUP, $MLDP$, $MLMP$ и $MLUP$ NP-трудны в сильном смысле.*

Доказательство ▷ Напомним определение NP-трудной в сильном смысле задачи о минимальном покрытии, введенной во второй главе. Пусть заданы множество $M = \{1, ..., m\}$ и набор его подмножеств $M_1, ..., M_n$ таких, что $\bigcup_{i \in N} M_i = M$, где $N = \{1, ..., n\}$. Совокупность $\widetilde{N} \subset N$ подмножеств $M_i, i \in \widetilde{N}$, называется покрытием множества M, если $\bigcup_{i \in \widetilde{N}} M_i = M$. Каждому M_i приписан единичный вес. Требуется найти покрытие минимального веса.

Сведем задачу о минимальном покрытии к задачам LDP, LMP и LUP, т.е. построим функции g и h такие, что функция g строит по входу t задачи о минимальном покрытии вход v задачи LDP (LMP, LUP), длина которого ограничена полиномом от длины $|t|$ входа t, за полиномиальное от $|t|$ время, а функция h строит по оптимальному решению задачи LDP (LMP, LUP) для входа v оптимальное решение задачи t за полиномиальное от $|t|$ время.

Сперва построим по произвольному входу задачи о минимальном покрытии вход задачи LDP (LMP, LUP), т.е. определим функцию g. Пусть $I := N$ – множество возможных мест открытия предприятий, $J := M$ – множество потребителей. Определим бюджет каждого потребителя $b_j := 2, j \in J$. Положим $c_{ij} := 1, i \in I, j \in M_i$. Иначе транспортные затраты положим равными 2, то есть сделаем оставшиеся транспортные пути закрытыми. И пусть $f_i := 1, i \in I$. Введем обозначение: $\widetilde{I} := \{i \in I | y_i = 1\}$. Докажем следующее вспомогательное утверждение:

Лемма 4 *Задачи LDP, LMP и LUP для входа $g(t)$ эквивалентны следующей задаче:*

$$|J| - |\widetilde{I}| \to \max_{\widetilde{I} \subseteq I}$$

при ограничении:

$$\bigcup_{i \in \widetilde{I}} M_i = J.$$

Доказательство ▷ Сперва покажем, что существуют оптимальные решения $(p^{LDP}, x^{LDP}, y^{LDP})$, $(p^{LMP}, x^{LMP}, y^{LMP})$ и $(p^{LUP}, x^{LUP}, y^{LUP})$ задач LDP, LMP и LUP для входа $g(t)$, в которых $p_{ij}^{LDP} = p_i^{LMP} = p^{LUP} = 1; \forall i \in I, j \in J$. Рассмотрим произвольное оптимальное решение $(p^{LDP}, x^{LDP}, y^{LDP})$ задачи LDP с входом $g(t)$ и набор $(\widetilde{p}^{LDP}, x^{LDP}, y^{LDP})$, где $\widetilde{p}_{ij}^{LDP} = 1; \forall i \in I, j \in J$. Проверив все ограничения, не трудно убедиться, что этот набор – допустимое решение. Из ограничений задачи LDP следует, что, если $x_{ij}^{LDP} = 1$, тогда $p_{ij}^{LDP} \leq 1$. Получаем следующее:

$$w_{LDP}(p^{LDP}, x^{LDP}, y^{LDP}) = \sum_{i \in I} \sum_{j \in J} p_{ij}^{LDP} x_{ij}^{LDP} - \sum_{i \in I} y_i^{LDP} \leq$$

$$\leq \sum_{i\in I}\sum_{j\in J} x_{ij}^{LDP} - \sum_{i\in I} y_i^{LDP} = w_{LDP}(\widetilde{p}^{LDP}, x^{LDP}, y^{LDP}),$$

где w_{LDP} – целевая функция задачи LDP. Откуда следует, что $(\widetilde{p}^{LDP}, x^{LDP}, y^{LDP})$ – оптимальное решение. Аналогично и для задач LMP и LUP.

Получается, что для нахождения оптимального решения задач LDP, LMP и LUP для входа $g(t)$ можно ограничится перебором допустимых решений вида $(1, \cdots, 1, x, y)$. При этом, если некоторый потребитель $j \in M_i$ не обслуживается, назначаем этого потребителя на предприятие i. Если это предприятие закрыто, то открываем его. Тем самым значение целевой функции не уменьшается. Получается, что данные частные случаи эквивалентны следующей задаче:

$$|J| - |\widetilde{I}| \to \max_{\widetilde{I}\subseteq I}$$

при ограничении:

$$\bigcup_{i\in\widetilde{I}} M_i = J.$$

Лемма доказана. $\triangleleft$

Согласно лемме 4 имеем, что максимум выше описанных частных случаев исследуемых задач достигается на $\tilde{I} = \tilde{N}^*$ – покрытии минимального веса, т.е. когда $\{i \in I : y_i = 1\} = \tilde{N}^*$.

Теперь, если мы дополним множество I произвольным множеством H ($|H| = |N|$), а множество J произвольным множеством Z ($|Z| = |N|$), и определим на новых множествах транспортные затраты и бюджет потребителей следующим образом: $c_{ij} = 1$, если $j = \alpha(i)$, для некоторой биекции $\alpha : H \mapsto Z$, и $c_{ij} = 2$ в противном случае; положим бюджеты новых потребителей равными 2. Тем самым получим вход задач $MLDP$,

$MLMP$ и $MLUP$. Аналогичным задачам LDP, LMP и LUP способом получим сведение задач $MLDP$, $MLMP$ и $MLUP$ к задаче о минимальном покрытии. Теорема доказана. $\triangleleft$

Доказать NP-трудность задачи LMP можно было еще проще. Для этого достаточно рассмотреть ее частный случай, когда все предприятия уже построены. Т.е., при условии $f_i = 0$. Сведение задачи о минимальном покрытии к этому частному случаю задачи LMP (задача MP) описано во второй главе.

Рассмотрим полный взвешенный граф K^{n+m}, в котором вершинами являются потребители и возможные места открытия предприятий, а веса ребер – транспортные затраты. Задачу LDP (LMP, LUP, $MLDP$, $MLMP$, $MLUP$) определенную на графе K^{n+m} обозначим как LDP^K (LMP^K, LUP^K, $MLDP^K$, $MLMP^K$, $MLUP^K$). Тогда из теоремы 8, если положить $c_{ik} = 2$, где либо $i, k \in I$, либо $i, k \in J$, следует утверждение:

Следствие 3 *Задачи LDP^K, LMP^K, LUP^K, $MLDP^K$, $MLMP^K$ и $MLUP^K$ NP-трудны в сильном смысле, даже если транспортные затраты удовлетворяют неравенству треугольника.*

Если перебирать все возможные размещения, и решать для каждого из них соответствующие задачи ценообразования, то мы получим точные методы решения исследуемых задач размещения производства и ценообразования. Тогда из теоремы 4 и полиномиальной разрешимости задач DP и UP получаем следующее утверждение:

Теорема 9 *Задачи LDP, LMP, LUP полиномиально разрешимы в случае фиксированного числа возможных мест открытия предприятий. А*

задачи $MLDP$, $MLMP$ и $MLUP$ полиномиально разрешимы в случае фиксированного числа открываемых предприятий.

Введем ряд обозначений, соответствующих произвольной оптимизационной задаче A с критерием максимизации целевой функции:

$L(A)$ – множество примеров задачи A. При этом произвольный пример $t \in L(A)$ будем называть задачей t;

$OPT_A(t)$ – оптимальное значение целевой функции в задаче $t \in L(A)$;

$D_A(t)$ – множество допустимых решений задачи $t \in L(A)$;

$F_A(t, s)$ – значение целевой функции в задаче $t \in L(A)$ на решении $s \in D_A(t)$.

Из теоремы 8 следует, что, при условии $P \neq NP$, нахождение оптимального решения исследуемых задач размещения и ценообразования с ростом размерности становится достаточно трудоемким процессом. Тогда имеет смысл рассмотреть вопрос нахождения "неплохого" допустимого решения. Обычно в этом случае рассматривают сложность задачи с точки зрения построения эффективного алгоритма нахождения приближенного решения с гарантированной оценкой точности, т.е. положение оптимизационной задачи в иерархии аппроксимационных классов [27].

При определении данной иерархии (см. 1 главу) погрешность решения $s \in D_A(t)$ задачи A с входом $t \in L(A)$ задаётся величиной:

$$R_A(t, s) = \max\left\{ \frac{F_A(t, s)}{OPT_A(t)}, \frac{OPT_A(t)}{F_A(t, s)} \right\} \geq 1.$$

Понятно, что если задача A является оптимизационной задачей с критерием максимизации целевой функции, то $R_A(t, s) = \frac{OPT_A(t)}{F_A(t,s)}$.

Во второй главе было показано, что при фиксированном размещении задачи LMP и $MLMP$ принадлежат классу $Log\text{-}APX$, а задачи

LDP, LUP, $MLDP$ и $MLUP$ полиномиально разрешимы. Далее рассмотрим лемму, в которой устанавливается в некотором смысле "ограничение сверху" на положение исследуемых задач в аппроксимационной иерархии:

Лемма 5 *Задачи LDP, LMP, LUP, $MLDP$, $MLMP$ и $MLUP$ принадлежат классу Poly-APX.*

Доказательство ▷ Рассмотрим частные случаи исследуемых задач. Обозначим за L^1DP, L^1MP и L^1UP задачи LDP, LMP и LUP соответственно, в которых открывается не более одного предприятия. Из теоремы 4 следует, что задачи L^1DP, L^1MP и L^1UP полиномиально разрешимы. Оптимальные решения этих задач и будем использовать в качестве допустимого решения исходных задач. Далее ограничимся рассмотрением только задачи LDP. Для задач LMP и LUP доказательство проводится аналогичным образом.

Очевидно, что произвольное оптимальное решение задачи L^1DP является допустимым решением задачи LDP. Покажем следующее:

$$OPT_{LDP}(t) \leq n * OPT_{L^1DP}(t),$$

где $n = |I|$. Пусть $(\tilde{y}, \tilde{p}, \tilde{x})$ произвольное допустимое решение задачи LDP. Введем следующие обозначения:

$$i^* = \arg \max_{i \in I: \tilde{y}_i = 1} \Big\{ \sum_j \tilde{p}_{ij} \tilde{x}_{ij} - f_i \Big\};$$

$$y_i^{i^*} = \begin{cases} 1, \text{если } i = i^*, \\ 0, \text{в противном случае.} \end{cases}$$

$$x_{ij}^{i^*} = \begin{cases} \tilde{x}_{ij}, \text{если } i = i^*, \\ 0, \text{в противном случае.} \end{cases}$$

Тогда решение $(y^{i^*}, \tilde{p}, x^{i^*})$ является допустимым решением задачи L^1DP. Кроме того, по определению, верно следующее соотношение:

$$F_{LDP}(t, (\tilde{y}, \tilde{p}, \tilde{x})) = \sum_{i,j} \tilde{p}_{ij}\tilde{x}_{ij} - \sum_{i} f_i\tilde{y}_i \leq n * (\sum_{i,j} \tilde{p}_{ij}x_{ij}^{i^*} - \sum_{i} f_i y_i^{i^*}) \leq$$

$$\leq n * OPT_{L^1DP}(t).$$

Получается, что оптимальное решение задачи L^1DP является допустимым решением задачи LDP с оценкой точности n.

По аналогии, если рассмотреть задачи ML^1DP, ML^1MP и ML^1UP, в которых открывается ровно одно предприятие, можно показать принадлежность задач $MLDP$, $MLMP$ и $MLUP$ к классу *Poly-APX*. Лемма доказана. $\triangleleft$

Следствие 4 *Задачи LDP^K, LMP^K, LUP^K, $MLDP^K$, $MLMP^K$ и $MLUP^K$ принадлежат классу Poly-APX, даже если транспортные затраты удовлетворяют неравенству треугольника.*

В лемме 5 представлены эффективные алгоритмы нахождения допустимых решений задач LDP, LMP и LUP с оценками точности, ограниченными полиномами от длины входа соответствующих задач. Но данный результат не гарантирует отсутствие полиномиальных алгоритмов с лучшими оценками, соответствующих ниже лежащим классам иерархии. Следующая теорема позволяет решить данную проблему.

Введем определение AP-сводимости [27]. Пусть задачи A и B – пара задач из класса NPO с критерием максимизации целевой функции.

Будем говорить, что задача A AP-сводится к задаче B тогда и только тогда, когда существуют две функции φ и ρ и положительная константа α такие, что:

- для любого $t \in L(A)$ и любого $r > 1$: $\varphi(t, r) \in L(B)$ и функция φ вычислима за полиномиальное время от длины $|t|$ входа t и r;
- для любого $t \in L(A)$, любого $r > 1$ и любого $s \in D_B(\varphi(t, r))$: $\rho(t, s, r) \in D_A(t)$ и ρ вычислима за полиномиальное время от $|t|$, длины решения s и r;
- для любого $t \in L(A)$, любого $r > 1$ и любого $s \in D_B(\varphi(t, r))$: из того, что $R_B(\varphi(t, r), s) \leq r$ следует, что $R_A(t, \rho(t, s, r)) \leq 1 + \alpha(r - 1)$.

Напомним определение замкнутого подкласса класса NPO относительно некоторой сводимости. Подкласс C класса NPO является замкнутым относительно сводимости Γ, если из Γ-сводимости задачи A к задаче $B \in C$ следует, что $A \in C$. Известно, что при условии $P \neq NP$ включения в аппроксимационной иерархии между классами строгие, а классы $FPTAS$, $PTAS$, APX, $Log\text{-}APX$ являются замкнутыми относительно AP-сводимости [27, 28, 33]. Ко всему прочему, учитывая транзитивность AP-сводимости, получается, что для любой $Poly\text{-}APX$-полной относительно AP-сводимости задачи при условии $P \neq NP$ не может существовать полиномиальных приближенных алгоритмов с лучшими оценками погрешности допустимых решений от оптимальных, чем оценки ограниченные полиномом от длины входа. Для исследуемых задач имеет место следующее утверждение:

Теорема 10 *Задачи LDP, LMP и LUP являются $Poly\text{-}APX$-полными относительно AP-сводимости.*

Доказательство ▷ Рассмотрим $Poly\text{-}APX$-полную относительно AP-сводимости задачу о максимальном независимом множестве [27, 28, 33]. Пусть задан произвольный граф $G = (V, E)$, где V – множество вершин, а $E \subseteq V \times V$ – множество ребер. Независимым множеством в графе G назовем подмножество вершин $\tilde{V} \subseteq V$ такое, что $\tilde{V} \times \tilde{V} \cap E = \emptyset$. Тогда задачу о максимальном независимом множестве (MIS) можно представить в виде следующей оптимизационной задачи:

$$|\tilde{V}| \to \max_{\tilde{V} \subseteq V}$$

при ограничении:

$$\tilde{V} \times \tilde{V} \cap E = \emptyset.$$

Далее рассмотрим сведение задачи MIS к задаче LDP, которое легко преобразовывается в сведение к задачам LMP и LUP. Будем строить по входу $t \in L(MIS)$ вход $\varphi(t) \in L(LDP)$ следующим образом. Определим $w(i) := |j \in V : (i, j) \in E|$, то есть $w(i)$ – число вершин, смежных с вершиной i. Для каждого $i \in I$ определим свой набор потребителей J_i такой, что $J_i \cap J_k = \emptyset, \forall k \in I \setminus i$, и $J_i \cap E = \emptyset$, а также $|J_i| = |I| - w(i) = n - w(i)$. Обозначим за $\varphi(t)$ вход, в котором $I := V$, $J := (\bigcup_{i \in I} J_i) \cup E$, затраты на открытие любого предприятия равны $n - 1$, бюджет у всех потребителей равен 2, а транспортные затраты определяются следующим образом: на предприятии i транспортные затраты $c_{ij} := 1$, если $j \in J_i$ или

$j = (i, k) \in E$, для некоторого $k \in I$, иначе транспортные затраты равны 2.

Пусть (y, p, x) – допустимое решение задачи LDP. Применим следующий алгоритм, который по решению (y, p, x) строит вспомогательное допустимое решение $(\tilde{y}, \tilde{p}, \tilde{x})$ задачи LDP, по которому легко строится решение задачи MIS. Положим $\tilde{p}_{ij} := 1$, при всех $i \in I$ и $j \in J$, а также $\tilde{y} := y$ и $\tilde{x} := x$. Просмотрим поочередно все предприятия $i \in I$. Если $\tilde{y}_i = 1$ и предприятие i не приносит прибыль, т.е. $\sum_{j \in J} \tilde{p}_{ij}\tilde{x}_{ij} - f_i \leq 0$, тогда положим $\tilde{y}_i := 0$ и $\tilde{x}_{ij} := 0$, для всех $j \in J$.

В силу выбора цен и входных данных решение $(\tilde{y}, \tilde{p}, \tilde{x})$ является допустимым. Также очевидно, что нет никакой разницы к какой именно задаче размещения и ценообразования относится это решение. Также отметим, что множество $\tilde{V} = \{i \in I : \tilde{y}_i = 1\}$ – независимое множество, т.е. допустимое решение задачи MIS. Действительно, если существует пара отличных друг от друга предприятий $i, k \in \tilde{V}$, тогда, т.к. потребитель (i, k) может обслуживаться только одним предприятием (для определенности будем считать, что предприятие i его не обслуживает), прибыль, приносимая предприятием i, равна:

$$\sum_{j \in J} \tilde{p}_{ij}\tilde{x}_{ij} - f_i = \sum_{j \in J \setminus (i,k)} \tilde{p}_{ij}\tilde{x}_{ij} - f_i \leq \sum_{s=\overline{1,n-1}} (1) - (n-1) = 0,$$

в силу определения транспортных затрат и затрат на открытие предприятия. Откуда следует, что предприятие $i \notin \tilde{V}$. Получается, что $\tilde{V}$ – независимое множество. Трудоемкость выше описанного алгоритма $O(n^2)$, а значит полиномиальна от длины входа задачи MIS. Тем самым мы определили функцию ρ из определения AP-сводимости. Положим $\alpha = 1$. Чтобы выше описанное сведение задачи MIS к задаче LDP было AP-

сведением остается показать следующее:

$$\text{если } R_B(\varphi(t,r),s) \le r, \text{ то } R_A(t,\rho(t,s,r)) \le 1+\alpha(r-1)$$

По построению решения $(\tilde{y},\tilde{p},\tilde{x})$ выполнено следующее неравенство:

$$F_{LDP}(\varphi(t),(y,p,x)) \le F_{LDP}(\varphi(t),(\tilde{y},\tilde{p},\tilde{x})).$$

Оптимальное значение задачи LDP равно $n|V^*|-(n-1)|V^*| = |V^*|$, где V^* – оптимальное решение задачи MIS. Таким образом, получим:

$$r \ge R_{LDP}(\varphi(t),(y,p,x)) \ge \frac{OPT_{LDP}(t)}{F_{LDP}(\varphi(t),(\tilde{y},\tilde{p},\tilde{x}))} =$$

$$= \frac{n|V^*|-(n-1/n)|V^*|}{n|\tilde{V}|-(n-1/n)|\tilde{V}|} = \frac{|V^*|}{|\tilde{V}|}.$$

Учитывая, что $\alpha = 1$, получим требуемое неравенство:

$$R_{MIS}(t,\rho(t,(y,p,x))) = \frac{|V^*|}{|\tilde{V}|} \le r.$$

Аналогично и для задач LMP и LUP.

Получается, что любая задача из класса $Poly\text{-}APX$ AP-сводится к исследуемым задачам размещения и ценообразования, а из леммы 5 следует их принадлежность к классу $Poly\text{-}APX$. Теорема доказана. $\triangleleft$

Из теоремы 10 и следствия 3, если положить $c_{ik} = 2$, где либо $i,k \in I$, либо $i,k \in J$, следует утверждение:

Следствие 5 *Задачи LDP^K, LMP^K и LUP^K являются Poly-APX-полными относительно AP-сводимости, даже если транспортные затраты удовлетворяют неравенству треугольника.*

3.2 Гибридные алгоритмы для задачи размещения и фабричного ценообразования

Учитывая сложность задачи $MLMP$, невозможность построения точного полиномиального алгоритма при условии $P \neq NP$ и отсутствие полиномиального приближенного метода с относительной погрешность лучше полиномиальной, были разработаны и исследованы два гибридных алгоритма, основанных на локальном поиске: спуск по чередующимся окрестностям (VND) и генетический локальный поиск (GLS) [37, 58, 64].

В [43] для решения задачи $MLMP$ предлагались двухэтапные эвристики, основанные на двух следующих идеях:

– сначала применяется локальный поиск по переменным y_i для выбора размещения r предприятий при фиксированным векторе цен p_i,

– затем локальный поиск применяется для выбора вектора цен p_i при заданном размещении предприятий.

При таком подходе возникают две подзадачи, каждая из которых проще исходной задачи. В [43] для решения подзадачи ценообразования использовалась VNS метаэвристика [58] с окрестностями N_k, $k = 1, \cdots, K$. В окрестности N_k, $k = 1, \cdots, K$ соседом вектора цен p_i является любой вектор, который получается изменением не более k компонент вектора p_i, соответствующих открытым предприятиям. Для решения подзадачи размещения предприятий применялись две метаэвристики: метод имитации отжига и метод чередующихся окрестностей (VNS метаэвристика) [43]. Соседние решения для текущего вектора y_i порождались с помощью двух окрестностей. Первая окрестность – окрестность k-swap, а вторая окрестность – окрестность $Lin–Kernighan$ [63]. В окрестности k-swap

любой сосед текущего размещения y_i может быть получен с помощью следующей процедуры: открываем $k^{'} \leqslant r$ новых предприятий и закрываем ровно столько же ранее открытых. Пусть $S = \{i|y_i = 1\}$ и χ_S – характеристическая функция множества S. Очевидно, что $y_i = \chi_S(i)$, $i = 1, \ldots, m$. Далее для упрощения изложения будем отождествлять булев вектор и соответствующую характеристическую функцию. Окрестность *Lin–Kernighan* для вектора y_i строится с помощью следующей итерационной процедуры:

1. Выбираем пару элементов $i_1 \in S, i_2 \in \overline{S}$ таких, что значение функции $f(\chi_{S\cup\{i_2\}\setminus\{i_1\}}, p)$ является максимальным среди всех таких пар. Если таких пар несколько, то выбирается любая из них.

2. Выбрасываем из множества S элемент i_1 и добавляем элемент i_2.

3. Повторяем шаги 1 и 2 k раз. При этом на каждой итерации в качестве индексов $i_1 \in S, i_2 \in \overline{S}$ можно использовать только такие, которые не возникали на предыдущих итерациях.

Эта процедура определяет k соседей текущего размещения y_i. Параметр k выбирается из интервала $[\min(r, m-r), \max(r, m-r)]$. Локальный поиск относительно этой окрестности используется в методе имитации отжига каждый раз, когда температура уменьшается, а в VNS метаэвристике когда рекордное решение не может быть улучшено в течении некоторого количества итераций.

Так как отыскание наилучшего соседа текущего размещения относительно любой из этих окрестностей оказывается трудоёмкой процедурой для больших k, то для VNS метаэвристики использовались окрестности с небольшими $k(k \leq 3)$ и с $k = 1$ для метода имитации отжига [43].

Приведем описание окрестностных структур, которые используются в гибридных алгоритмах.

Обозначим через $Flip(p \mid y)$ окрестность вектора цен p для заданного размещения y. Эта окрестность состоит из всех векторов, каждый из которых отличается от p ровно одной компонентой. Если $\overline{p} \in Flip(p \mid y)$ и для некоторого k мы имеем $\overline{p}_i = p_i, i \neq k$, то для вычисления значения целевой функции $f(y, \overline{p})$ и значения компоненты $\overline{p}_k$ решается задача $MLMP$ с заданным размещением y, ценами $\overline{p}_i, i \neq k$, и одной переменной p_k. Эта задача разрешима за полиномиальное время, как следствие теоремы 1 о необходимых условиях оптимальности задачи MP.

Окрестность $Swap(y \mid p)$ определяется для заданного вектора цен p. Она содержит все векторы $\overline{y}$, где $\sum \overline{y}_i = r$, с расстоянием Хэмминга от $\overline{y}$ до y равным 2. Рассмотрим такой вектор $\overline{y}$. Тогда для некоторых i_0 и i_1 таких, что $y_{i_0} = 0$, $y_{i_1} = 1$, выполняются равенства $\overline{y}_i = y_i, i \neq i_0, i_1$, $\overline{y}_{i_0} = 1$, $\overline{y}_{i_1} = 0$. Так как одно предприятие закрылось и появилось новое, то необходимо пересчитать цены с индексами i_0 и i_1. Пусть $\overline{p}_i = p_i, i \neq i_0, i_1$, $\overline{p}_{i_1} = 0$, а значение компоненты $\overline{p}_{i_0}$ получится при решении задачи $MLMP$ с заданным размещением предприятий $\overline{y}$, ценами $\overline{p}_i, i \neq i_0$, и одной переменной p_{i_0}.

Обозначим как $LS^{pr}(p \mid y)$ алгоритм локального поиска относительно окрестности $Flip(p \mid y)$, а алгоритм локального поиска относительно окрестности $Swap(y \mid p)$, обозначим как $LS^{loc}(y \mid p)$.

Далее приведем описание двух метаэвристик, использующих алгоритм локального поиска $LS^{pr}(p \mid y)$. Для решения исследуемой задачи $MLMP$, мы используем генетический алгоритм локального поиска

(GLS), который интересен с теоретической и практической точек зрения. Он может рассматриваться как вариант Memetic алгоритма, который использует разные жадные стратегии и операторы кроссовера [64]. Генетический алгоритм локального поиска является итерационным методом. На каждой итерации, имеется набор локальных оптимумов относительно используемой окрестности. Этот набор называется популяцией. Он развивается в течение ряда итераций, пока не выполнится критерий остановки. Схема предлагаемой метаэвристики может быть представлена следующим образом.

Алгоритм GLS

Вход:

y — *булевый вектор, задающий размещение предприятий;*

I_{max} — *максимальное число итераций.*

Выход:

Наилучший среди элементов финальной популяции вектор цен p.

Шаг 0:

$i \leftarrow 0$. *Случайно порождаем начальную популяцию векторов цен. Применяем алгоритм* $LS^{pr}(p \mid y)$ *к каждому вектору* p *из популяции.*

Шаг 1:

Случайно выбираем два элемента популяции в качестве родителей. С помощью равномерного кроссовера создаем потомка p' *для выбранных родителей. Применяем локальный поиск к вектору цен* p' *и находим локальный оптимум* $p^* := LS^{pr}(p' \mid y)$.

Шаг 2:

Модифицировать популяцию. $i \leftarrow i + 1$. *Если* $i \leq I_{max}$, *то перейти*

на шаг 1, иначе Стоп.

Метод спуска с чередующимися окрестностями (VND) выполняет некоторое количество итераций с разными окрестностями до тех пор пока не будет получен локальный оптимум относительно всех используемых окрестностей. Пусть N_1, $N_2, \cdots, N_K$ множество из K окрестностных структур. Начиная с первой структуры N_1, VND-алгоритм выполняет шаги локального поиска пока это возможно. Далее локальный поиск продолжается из уже найденного локального оптимума относительно окрестностной структуры N_2. Если в процессе просмотра элементов данной структуры удается найти лучшее решение чем исходный оптимум, то VND-алгоритм возвращается к использованию структуры N_1, иначе алгоритм продолжает локальный поиск относительно окрестностной структуры N_3, и так далее. Таким образом, если алгортитм добрался до окрестностной структуры N_K и в процессе просмотра её элементов не удается улучшить текущее решение, то оно является локальным оптимумом относительно всех использованных окрестностных структур. В этом случае алгоритм прекращает свою работу. Окрестности обычно просматриваются в определенном порядке. Например, от самой маленькой по мощности окрестности, в которой процесс просмотра элементов окрестности является самым быстрым, до самой большой и самой трудоёмкой при реализации локального поиска.

Далее приводится описание реализации этой эвристики.

Алгоритм VND

Вход:

y — булевый вектор, задающий размещение предприятий;

I_{max} — *максимальное число итераций алгоритма;*

d — *число улучшений текущего решения, допускаемых в процессе работы процедуры Improve;*

k — *параметр окрестности* k-$Flip(p \mid y)$.

Выход:

Наилучший вектор p *среди найденных в процессе работы алгоритма векторов цен.*

Шаг 0:

$i \leftarrow 0$. *Случайно породить начальный вектор цен* p.

Шаг 1:

Применить локальный поиск к вектору цен p *и найти локальный оптимум* $p^* \leftarrow LS^{pr}(p \mid y)$.

Шаг 2:

$i \leftarrow i + 1$. *Если* $i > I_{max}$, *тогда стоп, иначе* $p \leftarrow Improve(k, p^* \mid y)$. *Если* $f(y, p) > f(y, p^*)$, *то перейти на шаг 1, иначе стоп.*

В предлагаемой реализации процедуры $Improve$ используются следующие окрестностные структуры: 2-$Flip(p \mid y)$, 3-$Flip(p \mid y)$. Для ускорения локального поиска по каждой из этих окрестностных структур используется d улучшенная стратегия поиска, идея которой заключается в том, что как только найдено ровно d улучшений в данной окрестности, то поиск прекращается [58].

В предлагаемых гибридных алгоритмах на каждой итерации к текущему решению (y, p) вначале прменяется локальный поиск относительно окрестности $Flip(p \mid y)$, и затем применяется локальный поиск относительно окрестности $Swap(y \mid \overline{p})$, где вектор цен $\overline{p}$ – результат поиска по

предыдущей окрестности. В качестве критерия остановки используется достижение заранее заданного числа итераций. Результатом работы алгоритма является наилучшее найденное решение. Процедура первого шага основывается на генетическом локальном поиске в первом алгоритме и на VND эвристике во втором алгоритме. Теперь перейдем к описанию гибридных метаэвристик. Пусть Y – множество всех векторов $\overline{y}$ таких, что $\sum \overline{y}_i = r$, а через L обозначим табу лист.

Алгоритм LS+GLS

Вход:

I_{max} — максимальное число итераций алгоритма;

Выход:

Наилучшее найденное решение.

Шаг 0:

Положим $i \leftarrow 0$, $L \leftarrow \emptyset$. Случайно порождаем начальное размещение y.

Шаг 1:

Если $i > I_{max}$, тогда стоп, иначе $L := L \cup \{y\}$, $i \leftarrow i + 1$, применяем метаэвристику GLS и найдём локальный оптимум p.

Шаг 2:

Найти локальный оптимум $y^* := LS^{loc}(y \mid p)$. Пусть p^* — вектор цен соответствующий размещению y^*. Если $f(y^*, p^*) \leq f(y, p)$, то тогда выбрать любое размещение из множества $y \in Y \setminus L$, иначе $y := y^*$, и перейти на шаг 1.

Чтобы получить алгоритм LS+VND необходимо заменить алгоритм GLS алгоритмом VND.

3.3 Вычислительный эксперимент

Тестирование гибридных алгоритмов проводилось на персональном компьютере с процессором Intel Core i7-3612QM и 4Gb оперативной памяти. Алгоритмы LS+GLS и LS+VND сравнивались с уже известными алгоритмами: двухуровневой эвристикой VNS+VNS [43] и методом ветвей и границ из известной библиотеки CPLEX. При тестировании использовались входы размерности $n = 40, 100$, $m = 100$, $r = 5$ из библиотеки "Дискретные задачи размещения" ("Discrete Location Problems") (таблица 3.1). Отдельно приводятся результаты сравнения алгоритмов LS+GLS и LS+VND (таблица 3.2) на этих же входах, но другой размерности $n = 40$, $m = 100$, $r = 10, 15$. В [43] помимо двухуровневой эвристики VNS+VNS исследовалась эвристика SA+VNS. Но, в связи с тем, что этот алгоритм показывает результаты не лучше, чем алгоритм VNS+VNS, и при этом требует существенно большее вычислительное время, здесь соответствующие результаты не воспроизводятся. В таблицах 3.1 и 3.2 первый столбец обозначает код каждого входа. Столбец $n(r)$ задает количество возможных мест открытия предприятий и число открываемых предприятий. Столбцы revenue, time, iter. задают наилучший найденный доход, требуемое вычислительное время и номер итерации, на которой было найдено наилучшее решение. Для метода ветвей границ из пакета CPLEX приводятся результаты только для входов наименьшей размерности. Процесс счета обрывался по истечении 24 часов и наилучшее найденное решение предъявлялось как результат работы алгоритма. Для входов большей размерности по истечении указанного времени алгоритм не успевал найти хотя бы одно допустимое решение.

Instance		VNS+VNS		CPLEX	LS+GLS			LS+VND		
	n(r)	revenue	time	revenue	revenue	iter.	time	revenue	iter.	time
1	40(5)	2245	45m	2226	2245	5	36s	2245	20	4s
2	40(5)	2259	51m	2259	2259	14	97s	2259	72	16s
3	40(5)	2019	41m	2019	2019	8	47s	1984	12	3s
4	40(5)	1533	42m	1508	1533	63	347s	**1552**	22	5s
5	40(5)	2386	46m	2313	2386	11	77s	2346	34	8s
6	40(5)	1960	60m	1949	1956	9	57s	**1987**	17	4s
7	40(5)	2179	60m	2142	2179	55	415s	2178	58	14s
8	40(5)	2139	51m	2139	2139	30	224s	**2140**	31	7s
9	40(5)	1895	59m	1877	**1904**	17	115s	**1900**	45	10s
10	40(5)	2209	37m	2209	2209	4	32s	**2252**	4	1s
11	100(5)	2235	1h 9m		2230	7	48s	2235	24	14s
12	100(5)	2240	3h 12m		2240	295	2015s	2233	31	18s
13	100(5)	1923	1h 19m		1923	16	107s	**1957**	13	8s
14	100(5)	2133	1h 48m		2133	466	3194s	2118	15	9s
15	100(5)	2099	1h 58m		2099	27	197s	**2153**	25	15s
16	100(5)	2237	1h 10m		2237	108	806s	2182	126	75s
17	100(5)	1888	1h 15m		**1893**	31	202s	**1921**	108	62s
18	100(5)	1825	2h 51m		1825	48	312s	**1871**	4	2s
19	100(5)	1767	1h 43m		1767	8	48s	1767	4	3s
20	100(5)	2363	1h 2m		**2368**	55	369s	**2368**	139	84s

Таблица 3.1: Сравнение алгоритмов LS+GLS, LS+VND, VNS+VNS и CPLEX

Instance		LS+GLS			LS+VND		
	n(r)	revenue	iter.	time	revenue	iter.	time
21	40(10)	**2650**	43	2157s	2626	421	635s
22	40(10)	2617	39	2266s	**2621**	182	282s
23	40(10)	**2351**	31	1601s	2327	73	106s
24	40(10)	1888	55	2305s	**1895**	78	112s
25	40(10)	**2827**	24	942s	2793	88	141s
26	40(10)	2261	53	2902s	**2289**	349	538s
27	40(10)	**2481**	15	498s	2480	528	855s
28	40(10)	**2447**	56	1842s	2443	192	311s
29	40(10)	2245	47	2629s	**2246**	758	1250s
30	40(10)	2497	58	3054s	**2596**	112	180s
31	40(15)	**2804**	26	3162s	2796	794	3413s
32	40(15)	2758	26	3635s	**2797**	346	1616s
33	40(15)	**2515**	13	1908s	2477	146	620s
34	40(15)	1952	29	3161s	**1983**	453	2046s
35	40(15)	**2908**	11	1881s	2870	9	45s
36	40(15)	2286	23	3129s	**2436**	25	114s
37	40(15)	2529	16	2144s	**2571**	288	1369s
38	40(15)	2532	8	545s	**2558**	302	1510s
39	40(15)	2355	6	630s	**2372**	659	3091s
40	40(15)	2424	24	3431s	**2698**	52	261s

Таблица 3.2: Сравнение алгоритмов LS+GLS и LS+VND

Как следует из таблицы 3.1, алгоритмы LS+GLS и LS+VND не уступают по качеству найденных решений эвристике VNS+VNS и методу ветвей и границ, но при этом затрачивают меньшее вычислительное время. И, как показывает таблица 3.2, метод LS+VND оказывается предпочтительней метода LS+GLS с точки зрения затрат вычислительного времени для больших r.

3.4 Основные результаты третьей главы

В этой главе рассмотрена задача размещения производства и ценообразования с фабричной, равномерной и дискриминационной стратегиями ценообразования, а также с двумя различными типами размещения производства. Показано, что данные задачи являются NP-трудными в сильном смысле и допускают построение полиномиальных алгоритмов с относительной погрешностью, ограниченной полиномом от длины входа. Предложены полиномиально разрешимые случаи этих задач, а также показано, что задачи LDP, LMP и LUP являются $Poly\text{-}APX$-полными относительно AP-сводимости, что гарантирует невозможность построения полиномиальных приближенных алгоритмов с логарифмической и константной относительными погрешностями, а также $PTAS$ и $FPTAS$, при условии $P \neq NP$.

Для задачи $MLMP$ разработаны и исследованы гибридные алгоритмы, основанные на идеях генетического локального поиска, поиска с чередующимися окрестностями и локального поиска. Вычислительный эксперимент показал их конкурентоспособность в сравнении с известными методами решения.

Глава 4

Задача конкурентного размещения и ценообразования

В этой главе диссертации рассматривается задача конкурентного размещения производства и ценообразования, которая формулируется в виде игры Штакельберга. Первым выбирает размещение своих предприятий лидер, а затем конкурент. Далее игра развивается в соответствии со сценарием модели ценовой конкуренции Бертрана. Полученные равновесные цены используются для раздела рынков. Рынок достаётся тому из конкурентов, который может предложить наименьшую цену. На каждом из монополизированных рынков игрок устанавливает свою монопольную цену. Доход игрока складывается из доходов, которые он получает со своих монополизированных рынков.

Ниже приводятся результаты о вычислительной сложности задачи конкурентного размещения производства и ценообразования, исходя из которых, излагаются результаты по разработанным методам ее решения, основанным на идеях альтернирующей метаэвристики. Помимо этого расширяется понятие аппроксимационной иерархии на второй уровень

относительно полиномиальной иерархии классов сложности задач распознавания для уточнения сложности исследуемой модели.

4.1 Вычислительная сложность

В данном разделе будем предполагать, что исходные данные задачи являются целочисленными. Напомним определение стандартной задачи распознавания, соответствующей оптимизационной задаче с критерием максимизации целевой функции. Сопоставим оптимизационной задаче A задачу распознавания $D(A)$, в которой входом является вход задачи A и произвольное рациональное число k. В задаче $D(A)$ надо решить, существует или нет допустимое решение со значением целевой функции, большим или равным k.

Имеет место следующее утверждение:

Теорема 11 *Задача CLP и задача конкурента $F(x)$ являются NP-трудными в сильном смысле.*

Доказательство ▷ Рассмотрим задачу о $(r|p)$ – центроиде, в которой участвуют два игрока: лидер и конкурент. Сперва лидер размещает свои p предприятий в конечном множестве возможных мест размещения, затем конкурент открывает свои r предприятий. Каждый клиент выбирает ближайшее к нему (в смысле транспортных затрат) открытое предприятие и приносит ему некоторый доход (независимо от предприятия). Причем из двух предприятий лидера и конкурента, в которых его затраты одинаковы, он предпочтет предприятие лидера. Утверждение теоремы следует из эквивалентности $D(F(x))$, и стандартной задачи распознава-

ния, соответствующей задаче конкурента для задачи о $(r|p)$ – центроиде [22, 38, 70], если положить $d_{ik} = d_{jk}$, для любых $i, j \in I; k \in K$. Задача CLP является NP-трудной в сильном смысле, так как частный случай задачи при $r = 0$ эквивалентен NP-трудной в сильном смысле задаче о p-медиане [38], в которой требуется разместить p предприятий в конечном множестве возможных мест размещения так, чтобы суммарные транспортные затраты обслуживаемых клиентов были минимальны при условии, что каждый клиент выбирает ближайшее к нему предприятие. Понятно, что если положить в задаче CLP $r = 0$ и $d_{ik} = C - c_{ik}$, для некоторого большого C, то получим эквивалентную p-медиане задачу. ◁

Также выполнено следующее утверждение:

Теорема 12 *Задача CLP является Σ_2^P-трудной.*

Доказательство ▷ Задача распознавания $D(CLP)$ принадлежит классу Σ_2^P, если использовать в качестве NP-оракула задачу распознавания $D(F(x))$. Сведем Σ_2^P-полную стандартную задачу распознавания, соответствующую конкурентной задаче о p-медиане (задаче о $(r|p)$ – центроиде), к задаче $D(CLP)$ как в предыдущей теореме. А Σ_2^P-трудность задачи CLP следует из Σ_2^P-полноты задачи $D(CLP)$. ◁

Для того чтобы сформулировать следующий результат, необходимо ввести новую иерархию аппроксимационных классов. Стандартная аппроксимационная иерархия состоит из следующих классов [27]:

$$PO \subseteq FPTAS \subseteq PTAS \subseteq APX \subseteq Log - APX \subseteq$$

$$\subseteq Poly - APX \subseteq Exp - APX \subseteq NPO.$$

Каждый из этих классов описывает определённое качество аппроксимации, которым обладают образующие его оптимизационные задачи. Данная иерархия используется для описания свойств задач из класса NPO. Содержательно его можно описать как класс оптимизационных задач, у которых соответствующая задача распознавания принадлежит классу NP. Так как у задачи CLP, степень аппроксимируемости которой собираемся оценить, соответствующая задача распознавания является Σ_2^P-полной, то требуемое обобщение естественно ввести следующим образом:

$$\Delta_2^P O \subseteq FPTAS_2^P \subseteq PTAS_2^P \subseteq APX_2^P \subseteq Log - APX_2^P \subseteq$$

$$\subseteq Poly - APX_2^P \subseteq Exp - APX_2^P \subseteq \Sigma_2^P O,$$

где $\Sigma_2^P O$ – класс оптимизационных задач, у которых соответствующая задача распознавания принадлежит классу Σ_2^P. Определение любого из приведённых аппроксимационных классов получается из определения исходного базового класса заменой полиномиального алгоритма на полиномиальный оракульный алгоритм с оракулом из класса NP.

Таким образом, оптимизационная задача из класса $\Sigma_2^P O$ принадлежит к классу $Poly - APX_2^P$ тогда и только тогда, когда существует полиномиальный детерминированный оракульный алгоритм с подходящим оракулом из класса NP, находящий $p(r)$-приближенное решение задачи, где r – длина входа задачи, p – полином [27].

Теорема 13 *Задача CLP принадлежит классу $Poly - APX_2^P$.*

Доказательство ▷ Пусть $(i^*, k^*) = \arg\max_{(i,k)\in I\times K} d_{ik}$. Возьмем произвольное размещение $\overline{x}$ такое, что $\overline{x}_{i^*} = 1$, и рассмотрим допустимое решение $(\overline{x}, \overline{y}, \overline{z}^L, \overline{z}^F)$ задачи CLP с данным $\overline{x}$. Возможно два случая:

1) $\overline{z}^L_{i^*k^*} = 1$. Тогда выполняется неравенство

$$\sum_{k\in K}\sum_{i\in I} d_{ik}\overline{z}^L_{ik} \geqslant d_{i^*k^*}.$$

.

2) $\overline{z}^L_{i^*k^*} = 0$. В силу выбора пары (i^*, k^*) выполнено неравенство $d_{i^*k^*} \geqslant d_{ik^*}$ при всех $i \in I$. Отсюда следует, что $c_{ik^*} \geqslant c_{i^*k^*}$ при всех $i \in I$. Тогда получаем, что $I_k = \emptyset$. Это означает, что $\exists i^{'} \in I : \overline{z}^L_{i^{'}k^*} = 1$. Из ограничения (1.16) задачи конкурента следует, что $c_{i^{'}k^*} = c_{i^*k^*}$. В итоге получаем следующее: $d_{i^{'}k^*} = d_{i^*k^*}$ и выполняется неравенство

$$\sum_{k\in K}\sum_{i\in I} d_{ik}\overline{z}^L_{ik} \geqslant d_{i^*k^*}.$$

Пусть $(\tilde{x}, \tilde{y}, \tilde{z}^L, \tilde{z}^F)$ – оптимальное решение задачи CLP. Тогда согласно ограничению (1.14) оптимальное значение есть

$$\sum_{k\in K}\sum_{i\in I} d_{ik}\tilde{z}^L_{ik} \leqslant |K| \max_{(i,k)\in I\times K} d_{ik} = |K| d_{i^*k^*}.$$

С другой стороны, $d_{i^*k^*} \leqslant \sum\limits_{k\in K}\sum\limits_{i\in I} d_{ik}\overline{z}^L_{ik}$. Получаем, что

$$\sum_{k\in K}\sum_{i\in I} d_{ik}\tilde{z}^L_{ik} \leqslant |K| \sum_{k\in K}\sum_{i\in I} d_{ik}\overline{z}^L_{ik}.$$

Для нахождения допустимого решения $(\overline{x}, \overline{y}, \overline{z}^L, \overline{z}^F)$ задачи CLP с заданным размещением $\overline{x}$ необходимо решить задачу конкурента $F(x)$. Рассмотрим задачу $D(F(x))$. Очевидно, что значение целевой функции задачи конкурента лежит в целочисленном отрезке $[0, \sum_{k\in K} \max_{i\in I} d_{ik}]$. Получается, что, используя двоичный поиск, оптимальное решение задачи конкурента можно найти за полиномиальное от длины входа число обращений к задаче $D(F(x))$. Таким образом, показано, что предложенное выше $|K|$-приближенное решение задачи CLP можно найти с помо-

щью детерминированного полиномиального алгоритма, использующего оракул из класса NP. $\triangleleft$

Результаты, полученные в данном разделе, характеризуют сложность нахождения точных и приближенных решений исследуемой задачи. Из теоремы 12 следует, что при условии $NP \neq co\text{-}NP$ для задачи CLP не существует детерминированного алгоритма с NP-оракулом, находящего точное решение задачи за полиномиальное от длины входа время. Из теоремы 13 следует, что для задачи CLP существует полиномиальный алгоритм, находящий приближенное решение с полиномиальной от длины входа оценкой точности, использующий оракул из класса NP. Из доказательства этого результата следует, что этот подход требует решения NP-трудной задачи конкурента, а это значительно сказывается на его трудоемкости. Поэтому в следующем разделе предложены иные способы поиска приближенных решений, основанные на идеях построения приближенных решений для Σ_2^P-трудной задачи о $(r|p)$ – центроиде [29, 31].

4.2 Приближённые алгоритмы решения

Метаэвристики широко применяются для решения трудных задач комбинаторной оптимизации. Их главное преимущество связано с быстрой адаптацией к сложным математическим моделям, например к задачам размещения, ценообразования (как показано во второй и третьей главах работы), к задачам теории расписаний и др.

В этом разделе описываются два приближённых алгоритма для решения задачи CLP. Оба алгоритма основаны на идее альтернирующей

эвристики [29, 62]. В первом из них для заданного решения лидера решается задача конкурента и находится его наилучшее решение. После этого лидер выступает в роли конкурента, т.е. он ищет новое размещение своих предприятий при фиксированном решении конкурента. При этом множество $I_k(x)$ в задаче $F(x)$ необходимо заменить на множество вида

$$I_k(y) = \{i : c_{ik} \leq \min_{l:y_l=1} c_{lk}\}.$$

Множество $I_k(y)$ состоит из пунктов размещения, которые может использовать лидер для захвата k-го рынка. Этот процесс повторяется до тех пор, пока не выполнятся условия остановки. Другими словами, игроки по очереди решают задачу конкурента. Для равновесных задач сходимость аналогичных альтернирующих алгоритмов была установлена в [77]. Приведём схему первого алгоритма.

Алгоритм 3 (Альтернирующий)

Шаг 1 *Построить начальное решение. Положить наилучшее значение целевой функции (рекорд) равным нулю. Перейти на* **шаг 3***;*

Шаг 2 *Найти новое размещение лидера, решив внутреннюю задачу $F(y)$, где в качестве параметра выступает найденное на шаге 3 размещение конкурента. Перейти на* **шаг 3***;*

Шаг 3 *Решить задачу конкурента для текущего решения лидера. Если новое значение целевой функции задачи CLP больше, чем текущий рекорд, то запоминаем его. Если число итераций не превосходит заданной величины, то перейти на* **шаг 2**.

Наилучшее найденное решение является результатом работы алгоритма. В качестве начального решения лидера выбирается случайным обра-

зом сгенерированное множество из w точек.

Так как задача конкурента является труднорешаемой, то альтернирующий алгоритм требует значительных ресурсов для своей реализации. Поэтому рассмотрим далее гибридный альтернирующий алгоритм, в котором на шаге 2 вместо решения задачи $F(y)$ находится локальный оптимум относительно окрестности $Swap$. Окрестность $Swap(x, y)$ образуют все вектора (x', y'), полученные следующим образом: вектор x' образуется заменой одного элемента $i \in supp(x)$ из носителя вектора x на элемент из дополнения $j \in I \setminus supp(x)$. Если при этом $j \notin supp(y)$, то $y' = y$. В противном случае y' получается заменой элемента j на элемент $k \in \{I \setminus (supp(x) \cup supp(y))\} \cup \{i\}$, который максимизирует доход конкурента при фиксированном решении лидера. Приведём схему гибридного алгоритма.

Алгоритм 4 (Гибридный)

Шаг 1 *Построить начальное размещение* (x_0, y_0) *предприятий лидера и конкурента. Положить наилучшее значение целевой функции (рекорд) равным нулю;*

Шаг 2 *Найти локальный максимум для задачи размещения предприятий лидера по окрестности* $Swap(x, y)$*;*

Шаг 3 *Решить задачу конкурента для текущего решения лидера* x*. Если новое значение целевой функции задачи* CLP *больше, чем текущий рекорд, то поменять рекорд. Если число итераций не превосходит заданной величины, то перейти на* **шаг 2**.

Наилучшее найденное решение является результатом работы алгоритма. В качестве начального размещения предприятий лидера и конку-

рента выбираются два случайным образом сгенерированные множества соответственно из w и r точек.

4.3 Вычислительный эксперимент

Вычислительный эксперимент проводился на ЭВМ с процессором Core i7 3612QM. Алгоритмы тестировались на небольших по размерности примерах с числом возможных мест открытия предприятий, равным 20, и с 50 рынками, в которых значения величин d_{ik} и c_{ik} выбирались равновероятно из целочисленного интервала [0, 100]. Величины r и w брались равными друг другу и принимали значения 2, 3, 4, образуя тем самым 3 группы по 10 примеров. Результаты тестирования приведены в таблице 4.1, где первый столбец – номер примера ("Num"), в первой строке столбцы 2 – 4 обозначают группу примеров ($r = w = 2, 3, 4$), во второй строке в каждой группе стобцы "Alt" и "Hyb" обозначают результаты тестирования алгоритма 3 и алгоритма 4 соответственно.

В третьей строке поля "value" и "time" обозначают значение целевой функции на наилучшем из найденных решений и момент времени, когда это решение появляется в процессе работы соответствующего алгоритма. В данном эксперименте время измерялось в секундах. Из таблицы 4.1 видно, что в среднем алгоритм 3 работает медленнее и получает решения хуже, чем алгоритм 4. Однако есть примеры, для которых первый алгоритм находит решение, значительно превосходящее решение второго алгоритма, хотя на некоторых других примерах ситуация прямо противоположная. Во многом это связано с быстрым зацикливанием этих схем, когда получаемые решения начинают повторятся [21]. Альтернирующий

	r = w = 2				r = w = 3				r = w = 4			
Num	Alt		Hyb		Alt		Hyb		Alt		Hyb	
	value	time	value	time	value	time	value	time	value	time	value	time
1	2206	54	**2332**	22	2382	76	2382	22	2325	32	**2407**	23
2	2067	54	2067	44	2211	96	**2265**	107	**2372**	118	2091	22
3	1874	54	**2304**	32	2511	138	**2532**	65	2389	138	**2657**	56
4	2388	75	2388	32	2426	74	**2536**	32	**2464**	96	2657	55
5	2140	32	**2438**	54	**2621**	118	2317	22	2081	75	**2799**	34
6	1988	54	**2356**	87	2126	119	**2499**	32	**2668**	139	2651	67
7	2011	97	**2060**	55	2177	53	**2277**	53	2237	53	**2477**	22
8	1979	140	**2162**	23	2150	75	**2273**	161	2257	140	**2359**	145
9	**2239**	54	2135	57	2313	140	**2397**	65	**2552**	139	2444	79
10	2180	75	**2235**	68	2337	203	**2382**	32	**2445**	202	2250	34

Таблица 4.1: Сравнение алгоритмов 3 и 4.

алгоритм в среднем зацикливается быстрее, чем гибридный алгоритм. Таким образом, процедура локального улучшения размещения предприятий лидера относительно окрестности $Swap$ способствует увеличению длины циклов и, следовательно, увеличивает вероятность того, что появится решение с лучшим значением целевой функции. Также понятно, что с вычислительной точки зрения локальные преобразования являются менее трудоёмкими, чем решение задачи конкурента. Поэтому второй алгоритм оказывается более быстрым. Поиск методов, предотвращающих зацикливание, – весьма актуальная задача при разработке новых эффективных алгоритмов для решения рассмотренных задач [21].

4.4 Основные результаты четвёртой главы

В этой главе работы рассмотрена модель конкурентного размещения предприятий и ценообразования. Исследована вычислительная сложность задачи. Установлено, что она является Σ_2^P-трудной и лежит в классе $Poly\text{-}APX_2^P$. Для ее решения разработаны и исследованы алгоритмы, основывающиеся на идеях альтернирующей эвристики и локального поиска, основной недостаток которых связан с быстрым зацикливанием.

Глава 5

Задача государственно-частного партнерства

В данной главе идеи, связанные с теорией сложности и разработкой быстрых приближённых алгоритмов, применяются для изучения интересной и важной с прикладной точки зрения модели формирования работоспособного механизма государственно-частного партнерства, которая основывается на идеях Хотеллинга и механизме дисконтирования доходов и расходов, полученных и произведённых в разные моменты времени [62]. Решение такой задачи позволяет построить эффективную программу освоения минерально-сырьевой базы региона и может быть использовано для поддержки процесса принятия управленческих решений в природно-ресурсной сфере [10, 11].

Здесь показано, что исследуемая двухуровневая задача является NPO-трудной, при этом параметрическая задача нижнего уровня является NPO-полной. Исходя из этого, для решения модели государственно-частного партнерства разработаны и исследованы приближенные алгоритмы, основанные на методе локального поиска.

5.1 Вычислительная сложность

Теорема 14 *Задача PPP является NPO-трудной.*

Доказательство ▷ Для упрощения доказательства воспользуемся частным случаем AP-сводимости, а именно полиномиальным сведением, сохраняющим аппроксимируемость. Покажем, что можно свести за полиномиальное время NPO-полную задачу линейного булевого программирования с критерием максимизации целевой функции (обозначим ее как $Max^{LP}_{\{0,1\}}$) к задаче PPP, сохраняя аппроксимируемость. Задача $Max^{LP}_{\{0,1\}}$ задаётся целочисленной матрицей $A \in Z^{m\times n}$ и векторами $b \in Z^m$ и $c \in Z^n_{\geq 0}$. Булев вектор $\mathcal{X}$ – допустимое решение задачи, если выполняются неравенства $A\mathcal{X} \leq b$. В данной задаче среди допустимых решений необходимо найти оптимальное, на котором достигает максимума целевая функция $\sum_{j=1}^{n} c_j\mathcal{X}_j$. Обозначим индексное множество ограничений задачи $Max^{LP}_{\{0,1\}}$ через M, а индексное множество её переменных – через N.

Теперь построим требуемое сведение задачи $Max^{LP}_{\{0,1\}}$ к задаче PPP. Сначала покажем, как по исходным данным задачи $Max^{LP}_{\{0,1\}}$ строятся соответствующие данные задачи PPP. Положим $T = |M| + 1$, где T – горизонт планирования, $I = J = K = N, DG = DI = 0$,

$$ZPP^t_i = EPP^t_i, ZPI^t_j = EPI^t_j, t \leq T, i \in I, j \in J,$$

$$CFP^t_i = DBP^t_i = VDI^t_j = EDE^t_k = ZE^t_k = ZPE^t_k = 0, t \leq T, k \in K,$$

$$ZI^t_j = a_{tj}, t \leq T-1, j \in J, ZI^T_j = \sum_{t\leq T-1}(-a_{tj}) - c_j, j \in J,$$

$$b^G_t = b_t, t \leq T-1, b^G_T = -\sum_{(i,t):a_{ti}<0, t\leq T-1} a_{ti}.$$

$$\mu_{ij} = 0, i \in I, j \in J, \nu_{ik} = 0, i \in I, k \in K,$$

$$b_t^O = 0, t \leq T.$$

Теперь рассмотрим произвольное допустимое решение $\mathcal{X}$ задачи $Max_{\{0,1\}}^{LP}$ и покажем, что в задаче PPP для указанных выше исходных данных существует допустимое решение $(\overline{x}, \overline{y})$ с тем же значением целевой функции. Выберем значения переменных $\overline{y}_k = 0, k \in K$, и переменных $\overline{x}_j = \mathcal{X}_j, j \in N$.

Из вида исходных данных следует, что ограничения (1.21) для $t \leq T-1$ эквивалентны неравенствам

$$\sum_j ZI_j^t \overline{x}_j = \sum_j a_{tj} \mathcal{X}_j \leq b_t,$$

и, следовательно, выполнимы. Проверим ограничение с номером T, которое записывается в виде

$$\sum_j ZI_j^t \overline{x}_j \leq b_T^G = - \sum_{(j,t): a_{tj} < 0, t \leq T-1} a_{tj}.$$

Из определения величин ZI_j^T, неотрицательности вектора c, булевости вектора $\overline{x}$ и определения вектора b_T^G имеем следующую цепочку равенств и неравенств

$$\sum_j ZI_j^T \overline{x}_j = \sum_j \sum_{t \leq T-1} (-a_{tj}) \overline{x}_j + \sum_j (-c_j) \overline{x}_j \leq \sum_{(j,t): a_{tj} > 0, t \leq T-1} (-a_{tj}) \overline{x}_j +$$

$$+ \sum_{(j,t): a_{tj} < 0, t \leq T-1} (-a_{tj}) \overline{x}_j \leq \sum_{(j,t): a_{tj} < 0, t \leq T-1} (-a_{tj}) = b_T^G.$$

Исходные данные задачи инвестора выбраны так, чтобы для пары векторов $(\overline{x}, \overline{y})$ в задаче всегда были оптимальные решения, которые совпадают с множеством всех булевых векторов вида (y, z). Таким образом, ограничение (1.22) выполняется всегда.

Покажем, что значения целевых функций рассматриваемых задач совпадают на соответствующих решениях.

$$\sum_{t\le T}\sum_{j} ZI_j^t x_j = \sum_{j}\sum_{t\le T-1}(-a_{tj})x_j + \sum_{j} ZI_j^T x_j = \sum_{j}\sum_{t\le T-1}(-a_{tj})x_j +$$

$$+\sum_{j}\sum_{t\le T-1} a_{tj}x_j + \sum_{j} c_j x_j = \sum_{j} c_j \mathcal{X}_j.$$

Таким образом, предлагаемая сводимость сохраняет аппроксимируемость. Поскольку вопрос о принадлежности стандартной задачи распознавания $D(PPP)$ классу NP открыт, то задача государства является NP-трудной. И, следовательно, задача государства является NPO-трудной.

Теорема 14 доказана. $\triangleleft$

Теорема 15 *Задача инвестора $F_{PI}(x,y)$ является NPO-полной.*

Доказательство $\triangleright$ Покажем, как и в предыдущей теореме, что можно свести NPO-полную задачу $Max^{LP}_{\{0,1\}}$ за полиномиальное время к задаче $F_{PI}(x,y)$, сохраняя аппроксимируемость. Сначала покажем, как по исходным данным задачи $Max^{LP}_{\{0,1\}}$ строятся соответствующие данные задачи $F_{PI}(x,y)$. Положим $T = |M| + 1$, где T – горизонт планирования, $I = J = K = N, DI = 0,$

$$\mu_{ij} = 0, i \in I, j \in J, \nu_{ik} = 0, i \in I, k \in K,$$

$$ZPP_i^t = EPP_i^t, ZPI_j^t = EPI_j^t, t \le T, i \in I, j \in J,$$

$$EDE_k^t = ZE_k^t = ZPE_k^t = 0, t \le T, k \in K,$$

$$CFP_i^t = -a_{ti}, t \le T-1, i \in I, CFP_i^T = \sum_{t\le T-1} a_{ti} + c_i, i \in I,$$

$$b_t^O = b_t, t \le T-1, b_T^O = -\sum_{(i,t):a_{ti}<0, t\le T-1} a_{ti}.$$

Теперь рассмотрим произвольное допустимое решение $\mathcal{X}$ задачи $Max^{LP}_{\{0,1\}}$ и покажем, что в задаче $F_{PI}(x,y)$ для указанных выше исходных данных существует допустимое решение (z,u) с тем же значением целевой функции. Выберем значения переменных $u_k = 0, k \in K$, а переменных $z_i = \mathcal{X}_i, i \in N$. Очевидно, что ограничения (1.25) – (1.27), (1.29) выполнены для любых значений булевых переменных x, y, z. Из вида исходных данных следует, что ограничения (1.28) для $t \le T-1$ эквивалентны неравенствам

$$-\sum_j CFP_j^t z_j = \sum_j a_{tj} x_j \le b_t,$$

и, следовательно, выполнимы. Осталось проверить ограничение с номером T, которое записывается в следующем виде

$$-\sum_j CFP_j^T z_j \le b_T^O = -\sum_{(i,t):a_{ti}<0, t\le T-1} a_{ti}.$$

Из определения величин CFP_i^T, b_T^O, булевости вектора z и того, что вектор c является неотрицательным, имеем следующую цепочку равенств и неравенств

$$-\sum_j CFP_j^T z_j = \sum_j \sum_{t\le T-1} (-a_{ti}) z_j + \sum_j (-c_j) z_j \le \sum_{(i,t):a_{ti}>0, t\le T-1} (-a_{ti}) z_j +$$

$$+ \sum_{(i,t):a_{ti}<0, t\le T-1} (-a_{ti}) z_j \le \sum_{(i,t):a_{ti}<0, t\le T-1} (-a_{ti}) = b_T^O.$$

Покажем, что значения целевых функций рассматриваемых задач совпадают на соответствующих решениях.

$$\sum_{t\le T}\sum_j CFP_j^t z_j = \sum_j \sum_{t\le T-1} (-a_{ti}) z_j + \sum_j CFP_j^T z_j = \sum_j \sum_{t\le T-1} (-a_{ti}) z_j +$$

$$+\sum_j \sum_{t\le T-1} a_{ti} z_j + \sum_j c_j z_j = \sum_j c_j \mathcal{X}_j \ge 0.$$

Из последнего неравенства также следует, что выполняется последнее из оставшихся ограничений (1.30).

Исходные данные задачи $F_{PI}(x, y)$ устроены так, что можно ограничиться допустимыми решениями вида $(z, 0)$. И поэтому несложно проверить, что для любого допустимого решения $(z, 0)$ задачи инвестора найдётся допустимое решение задачи $Max^{LP}_{\{0,1\}}$ со значениями $\mathcal{X}_i = z_i, i \in N$ и значения целевых функций задач на этих решениях также совпадают. Таким образом, предлагаемая сводимость сохраняет аппроксимируемость. Заметим, что соответствующая стандартная задача распознавания $D(F_{PI}(x, y))$ принадлежит классу NP для любых значений параметров x и y. Следовательно, задача инвестора является NPO-полной.

Теорема 15 доказана. $\triangleleft$

Полученные результаты не позволяют в полной мере оценить связи исследуемой задачи с полиномиальной и аппроксимационной иерархиями. Не хватает информации о положении стандартной задачи распознавания для задачи PPP в полиномиальной иерархии. Однако эти результаты уже сейчас позволяют сделать вывод о ее большой вычислительной сложности. Из них следует, что для решения задачи реальной размерности любым точным алгоритмом потребуются слишком большие вычислительные ресурсы, несмотря на все достижения в области разработки вычислительных систем. Также эти результаты говорят о том, что не имеет смысла затрачивать усилия на разработку полиномиальных приближенных алгоритмов с гарантированными оценками относительного уклонения от оптимума. По крайней мере без дополнительных предпо-

ложений на исходные данные, которые позволили бы получить соответствующие полиномиальные приближённые или точные алгоритмы.

5.2 Приближённые алгоритмы решения

Для решения задачи планирования государственно-частного партнерства разработан гибридный алгоритм, основанный на методе локального поиска. Так как исследуемая задача двухуровневая и произвольное допустимое решение (x, y, z^*, u^*) включает в себя оптимальное решение (z^*, u^*) параметрической задачи инвестора с параметрами x и y, то решение (x, y) будем называть допустимым решением, если оно удовлетворяет ограничениям (1.21) и (1.23), а задача инвестора $F_{PI}(x, y)$ разрешима.

Введем в качестве параметров алгоритма следующие обозначения: $mIter$ – максимальное число итераций алгоритма нахождения начального решения (Шаг 2), $cfBound$ – коэффициент ослабления ограничения на значение целевой функции (1.20) при решении вспомогательной задачи на Шаге 2.3.

Алгоритм 5 (Гибридный)

Шаг 1 *Вычислить верхнюю границу Bound, решив задачу государства с ограничениями нижнего уровня (т. е. задачу с целевой функцией (1.20) и ограничениями (1.21), (1.23), (1.25) – (1.31)).*

Шаг 2 *Найти допустимое решение* (x^0, y^0) *(которое далее будет использоваться как начальное решение алгоритма локального поиска):*

Шаг 2.1 $iter := 1$.

Шаг 2.2 *Если* $iter \leqslant mIter$*, тогда решить задачу инвестора с переменными и ограничениями государства (т. е. задачу с целевой функци-*

ей (1.24) и ограничениями (1.21), (1.23), (1.25) – (1.31)), а также с дополнительным ограничением, что целевая функция государства (1.20) не меньше величины $(Bound - 1)/iter$. *Иначе перейдем на Шаг 3.*

Шаг 2.3 *Если задача на предыдущем шаге разрешима, а* (x^*, y^*, z^*, u^*) *– оптимальное решение, тогда вычислим значение* f *целевой функции (1.20), решив задачу* $F_{PI}(x^*, y^*)$. *Если* $f < (Bound - 1)/(iter * cfBound)$ *или задача на предыдущем шаге была не разрешима, тогда положим* $iter := iter + 1$ *и перейдем на Шаг 2.2, иначе положим* $x^0 := x^*$, $y^0 := y^*$, $f^0 := f$ *и перейдем на Шаг 3.*

Шаг 3 *Если на Шаге 2 не удалось найти допустимое решение, то в качестве него будем использовать нулевое решение, т. е. положим* $x^0 := 0$, $y^0 := 0$ *и вычислим значение* f^0 *целевой функции (1.20), решив задачу* $F_{PI}(x^0, y^0)$. *Далее применим алгоритм локального поиска:*

Шаг 3.1 *Положить в качестве стартового решения* $(x^*, y^*) := (x^0, y^0)$, *а рекорд* $f^* := f^0$.

Шаг 3.2 *Найти наилучшего соседа* (x, y) *в окрестности решения* (x^*, y^*).

Шаг 3.3 *Если значение целевой функции* $f(x, y) > f^*$, *тогда положим* $x^* := x$, $y^* := y$, $f^* := f(x, y)$ *и перейдем на Шаг 3.2, иначе стоп.*

В качестве окрестности в алгоритме локального поиска использовались окрестности $move$ и $rand$. Окрестностью $move$ решения (x, y) назовем все такие решения $(\tilde{x}, \tilde{y})$, где $\tilde{x}$ и $\tilde{y}$ отличаются соответственно от x и y не более чем двумя компонентами, при этом пара инфраструктурных или экологических проектов не могут быть одновременно открыты либо закрыты. Рандомизированная окрестность $rand$ решения (x, y) име-

ет ровно одно решение $(\tilde{x}, \tilde{y})$, полученное следующим образом. Каждая компонента вектора $\tilde{x}$ является случайной величиной, которая с вероятностью $1 - 1/|J|$ равна соответствующей компоненте вектора x, а с вероятностью $1/|J|$ – вектора $1 - x$. Аналогично и для векторов $\tilde{y}$ и y, только вероятности соответственно равны $1 - 1/|K|$ и $1/|K|$. В процессе работы алгоритма локального поиска с рандомизированной окрестностью в качестве критерия остановки бралось ограничение на число итераций.

В зависимости от выбора окрестностей $move2$ или $rand$ в алгоритме локального поиска будем называть гибридный алгоритм $move$ или $rand$ соответственно.

5.3 Вычислительный эксперимент

Тестирование работы алгоритмов $move$ и $rand$ проводилось на примерах со случайными исходными данными (все величины были целочисленными и брались из отрезка [0, 100], за исключением бюджетов) на 8 группах примеров разной размерности (по 10 примеров в каждой группе) от 30 производственных, 10 инфраструктурных и 30 экологических проектов до 90 производственных, 30 инфраструктурных и 90 экологических проектов. Во всех примерах $|T| = 20$. Результаты эксперимента представлены в таблице 5.1. Здесь столбец 1 обозначает номер группы примеров, столбцы 2–5 – размерность задачи, в столбцах 6 (7) и 8 (9) – усредненная по всем примерам в группе относительная погрешность $(Bound - f^*)/Bound$ работы алгоритма (усредненное время счета) $move$ и $rand$ соответственно, где $Bound$ – верхняя граница, а f^* – наилучшее найденное решение.

Пример					move		rand	
№	|I|	|J|	|K|	|T|	Уклон.	Время	Уклон.	Время
1	30	10	30	20	0,866	3,3 с	0,514	69,7 с
2	30	30	30	30	0,306	8,1 с	0,306	82,3 с
3	30	30	30	30	0	2,2 с	0	2,2 с
4	40	40	40	40	0,258	20,3 с	0,266	182,1 с
5	50	20	50	20	0,287	22,7 с	0,267	207,4 с
6	50	50	100	10	0	58,9 с	0	58,9 с
7	70	30	70	20	0,203	359,8 с	0,206	884,3 с
8	90	30	90	20	0,245	3438,9 с	0,228	4751,9 с

Таблица 5.1: Результаты численных экспериментов.

Время работы локального спуска в алгоритме $rand$ ограничивалось 5000 итераций, а параметры $mIter$ и $cfBound$ равны 30 и 3 соответственно. В силу большого числа итераций рандомизированного алгоритма локального поиска алгоритм $rand$ работает несколько дольше, чем алгоритм $move$, но при этом качество найденных решений в среднем оказывается лучше.

5.4 Основные результаты пятой главы

В этой главе рассмотрена новая прикладная задача формирования оптимального механизма государственно-частного партнерства, возникающая при согласовании долгосрочных интересов государства, частных инвесторов и населения в природно-ресурсной сфере, которая формулируется в виде двухуровневой задачи булевого программирования. Показана её NPO-трудность. Для решения задачи разработаны и исследованы приближенные алгоритмы, основанные на идеях локального поиска.

Заключение

1) Предложены новые модели ценообразования, размещения и ценообразования, конкурентного размещения и ценообразования, государственно-частного партнерства.

2) Установлено, что задачи дискриминационного и равномерного ценообразования полиномиально разрешимы, а задача фабричного ценообразования NP-трудна в сильном смысле и принадлежит классу $Log\text{-}APX$. Также для последней задачи разработаны эффективные гибридные алгоритмы, основанные на методе декомпозиции и метаэвристиках.

3) Установлено, что рассматриваемые задачи размещения и ценообразования являются NP-трудными в сильном смысле и принадлежат классу $Poly\text{-}APX$, причем задачи, когда за размещение взимается плата, являются $Poly\text{-}APX$-полными относительно AP-сводимости, а это значит, что для них не существует эффективного алгоритма с погрешностью "лучше" полиномиальной при условии $P \neq NP$. Для задачи размещения и фабричного ценообразования, когда требуется открыть известное число предприятий, разработаны гибридные алгоритмы на основе генетического локального поиска и спуска с чередующимися окрестностями, которые по точности получаемых решений и трудоемкости превосходят все известные ранее приближенные алгоритмы на тестовых примерах из

библиотеки "Дискретные задачи размещения".

4) Установлено, что задача конкурентного размещения и ценообразования является Σ_2^P-трудной и лежит в классе $Poly\text{-}APX_2^P$. Для нее разработаны эффективные алгоритмы, основывающиеся на идеях альтернирующей эвристики и локального поиска.

5) Установлено, что задача государственно-частного партнерства является NPO-трудной. Для нее разработан эффективный гибридный алгоритм, основывающийся на методе локального поиска.

Литература

[1] Береснев В.Л. О задаче конкурентного последовательного размещения предприятий со свободным выбором поставщиков // АиТ. 2014. № 4.

[2] Береснев В. Л., Мельников А. А. Приближённые алгоритмы для задачи конкурентного размещения предприятий // Дискретн. анализ и исслед. операций. 2010. т. 17. № 6. С. 3 – 19

[3] Васильев Ф.П. Методы оптимизации. — М.: Факториал Пресс, 2002. — 829 с.

[4] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. Москва: Мир. 1982.

[5] Давыдов И.А. Локальный поиск с запретами для дискретной задачи о (r|p)-центроиде // Дискрет. анализ и исслед. операций. 2012. Т. 19. № 2. С. 19-40.

[6] Давыдов И.А., Кочетов Ю.А. , Младенович Н., Уросевич Д. Быстрые метаэвристики для дискретной задачи о (r|p)-центроиде // АиТ. 2014. № 4.

[7] Дементьев В.Т., Шамардин Ю.В. Задача о выборе цен на продукцию при условии обязательного удовлетворения спроса // Дискретный

анализ и исследование операций. — 2002. — Серия 2, Том 9, № 2. — С. 31–40.

[8] Кибзун А. И., Наумов А.В., Иванов С.В. Двухуровневая задача оптимизации деятельности железнодорожного транспортного узла // Управление большими системами. 2012. № 38. С. 140-160.

[9] Кочетов Ю.А., Плясунов А.В. Полиномиально разрешимый класс задач двухуровневого линейного программирования // Дискретный анализ и исследование операций. — 1997 — Серия 2, том 1, № 2. — С. 23-33.

[10] Лавлинский С.М. Государственно-частное партнерство на сырьевой территории - экологические проблемы, модели и перспективы // Проблемы прогнозирования. 2010. № 1. С. 99-111.

[11] Лавлинский С.М., Калгина И.С. О методах оценки механизма государственно-частного партнерства в минерально-сырьевой сфере Забайкальского края // Вестник ЗабГУ. 2012. № 9(88) С. 96-102.

[12] Мельников А.А. Рандомизированный локальный поиск для дискретной задачи конкурентного размещения предприятий // АиТ. 2014. № 4.

[13] Панин А.А. Генетический алгоритм для одной задачи ценообразования // Труды ИВМиМГ СО РАН серия Информатика. — Новосибирск: URSS, 2009. — Вып. 9. — С. 190–196.

[14] Плясунов А. Гибридные методы решения сложных комбинаторных задач, использующие декомпозицию // Сборник докладов 8-й меж-

дународной конференции "Интеллектуальная обработка информации". Республика Кипр, г. Пафос, 17-24 октября, 2010. — С. 286–289.

[15] Плясунов А.В. О вычислительных возможностях метаэвристик // Материалы Российской конференции "Дискретная оптимизация и исследование операций". Владивосток, 7–14 сентября. — Новосибирск: Издательство Института математики, 2007. — С. 284–285.

[16] Схрейвер А. Теория линейного и целочисленного программирования. // Москва: Мир. 1991.

[17] Франк Р.Х. Микроэкономика и поведение. М.: ИНФРА-М, 2000. с. 696

[18] Aboolian R., Berman O., Krass D. Competitive facility location and design problem // Eur. J. Oper. Res. 2007. V. 182. P. 40-62.

[19] Aboolian R., Berman O., Krass D. Competitive facility location model with concave demand // Eur. J. Opl. Res. — 2007, — V. 181, — P. 598—619.

[20] Aboolian R., Berman O. and Krass D. Optimizing pricing and location decisions for competitive service facilities charging uniform price // Journal of the Operational Research Society. — 2008, — V. 59, — P. 1506 –1519.

[21] Alekseeva E., Kochetov Yu. Matheuristics and exact methods for the discrete (r|p)-centroid problem In: El-G. Talbi, L.Brotcorne (Eds.)

Metaheuristics for Bi-level Optimization (Studies in Computational Intelligence). Springer, 2013.

[22] Alekseeva E.V., Kochetova N.A., Kochetov Yu.A., Plyasunov A.V. A heuristic and exact methods for the dicrete $(r|p)$-centroid problem // LNCS. — Berlin: Springer, 2010. — V. 6022. — P. 11–22.

[23] Alekseeva E., Kochetova N., Kochetov Y., Plyasunov A. A hybrid memetic algorithm for the competitive p-median problem // Preprints of the 13th IFAC Symposium on Information Control Problems in Manufacturing, Moscow, Russia, June 3 - 5, 2009. — P. 1516–1520.

[24] Alekseeva E., Kochetov Yu., Plyasunov A. An exact method for the discrete (r|p)-centroid problem // Journal of Global Optimization. 2013. DOI: 10.1007/s10898-013-0130-6.

[25] Attallah M. Algorithms and theory of computation handbook. Boca Raton: CRC Press LLC, 1999.

[26] Audet, C., Savard, G., Zghal, W. New Branch-and-Cut Algorithm for Bilevel Linear Programming // J. Optim. Theory Appl. 2007. Vol. 134. P. 353-370.

[27] Ausiello G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A., Protasi M. Complexity and approximation: combinatorial optimization problems and their aproximability properties. Berlin: Springer-Verlag, 1999.

[28] Bazgan C., Escoffer B., and Paschos V. Th. Completeness in standard and differential approximation classes: Poly-(D)APX- and (D)PTAS-completeness // Theoret. Comput. Sci., 2005, Vol. 339. P. 272-292

[29] Bhadury J., Eiselt H.A., Jaramillo J.H. An alternating heuristic for medianoid and centroid problems in the plane // Comput. Oper. Res. 2003. V. 30. P. 553 – 565.

[30] Bouhtou M., Grigoriev A., Van Der Kraaij A.F., Van Hoesel S., Spieksma F., Uetz M. Pricing bridges to cross a river // Naval Research Logistics. — 2007. — V. 54, No. 4. — P. 411–420.

[31] Carrizosa, E., Davydov, I. and Kochetov Yu. A new alternating heuristic for the (r|p)-centroid problem on the plane // Oper. Res. Proc. 2011. Springer, 2012. P. 275-280.

[32] Chandru V., Hooker J.N. Optimization Methods for Logical Inference. — New York: John Wiley & Sons, 1999. — 365 p.

[33] Crescenzi P., Kann V., Silvestri R., and Trevisan L. Structure in approximation classes // SIAM J. COMPUT., 1999, Vol. 28. No. 5, P. 1759-1782

[34] Dasci A., Laporte G. Location and pricing decisions of a multistore monopoly in a spatial market // J. Region Sci. — 2004, — V. 44, — P. 489—515.

[35] Daskin, M.S. Network and Discrete Location: Models, Algorithms, and Applications. New York: John Wiley & Sons, 1995.

[36] Davydov I., Kochetov Yu., Carrizosa E. VNS heuristic for the (r|p)-centroid problem on the plane // Electronic Notes in Discrete Mathematics. 2012. Vol. 39. P. 5-12.

[37] Davydov I., Kochetov Yu., Mladenovic N., Urosevic D. Fast metaheuristic for the discrete (r|p)-the centroid problem // Automation and Remote Control. — 2014. — Vol. 75, N.4. — P. 677–687

[38] Davydov I., Kochetov Yu., Plyasunov A. On the complexity of the (r|p)-centroid problem in the plane // TOP 2013. DOI:10.1007/s11750-013-0275-y.

[39] Dechter R. Constraint Processing. — San Francisco: Morgan Kaufmann Publishers, 2003. — 481 p.

[40] Dempe S. Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints // Optimization. 2003. Vol. 52, P. 333-359.

[41] Dempe S.J. Foundations of bilevel programming. — Dordrecht: Kluwer Academic Publishers, 2002. — 481 p.

[42] DeNegre, S. T., Ralphs, T. K. A Branch-and-cut Algorithm for Integer Bilevel Linear Programs // Operations Research/Computer Science Interfaces. 2009. Vol. 47. P. 65-78.

[43] Diakova Z., Kochetov Yu. A double VNS heuristic for the facility location and pricing problem // Electronic Notes in Discrete Mathematics. — 2012. — Vol. 39, — P. 29–34.

[44] Drezner Z. Facility Location. A Survey of Applications and Methods. — Berlin: Springer, 1995.

[45] Drezner T., Eiselt H.A. Consumers in competitive location models // In: Z. Drezner, H. Hamacher (Eds.) Facility Location. Applications and Theory. — Berlin: Springer, 2004. — P. 151–178.

[46] Eiselt H.A. Competition in locational models // Studies in Locational Analyses, 1993. Vol. 5. P. 129–147.

[47] Eiselt H.A., Laporte, G. Competitive spatial models // European Journal of Operational Research, 1989. Vol. 39. P. 231–242.

[48] Eiselt H.A., Laporte G. Sequential location problems // Eur. J. Oper. Res. 1996. V.96. P. 217–242.

[49] Eiselt H.A., Laporte G. Thisse J.-F. Competitive location models: a framework and bibliography // Transportat. Sci. 1993. V. 27. P. 44 – 54.

[50] Eiselt, H.A., Marianov, V. Foundations of Location Analysis. —New York: Springer, 2011.— 510 p.

[51] Fischetti M., Lodi A. Local branching // Math. Program. 2003. Ser. B. Vol. 98. P. 23-47.

[52] Garcia M.D., Fernandez P., Pelegrin B. On price competition in location-price models with spatially separated markets // TOP. 2004. V. 12. P. 351 – 374.

[53] Geoffrion A.V. Generalized Benders decomposition // Journal of Optimization Theory and Application. — 1972. — Vol. 10, No. 4. — P. 237–260.

[54] Gote G., Laughton M.A. Large scale mixed integer programming: Benders-type heuristics // European Journal of Operational Research. — 1984. — Vol. 16. — P. 327–333.

[55] Grigoriev A., van Loon J., Sviridenko M., Uetz M., and Vredeveld T. Optimal Bundle Pricing with Monotonicity Constraint // Operations Research Letters. — 2008. — V. 36, No.5. — P. 609-614.

[56] Hamacher H.W., Nickel S. Classification of location models // Locat. Sci. 1998. V. 6. P. 229 – 242.

[57] Hanjoul P., Hansen P., Peeters D. and Thisse J-F. Uncapacitated plant location under alternative spatial price policies // Market Sci. — 1990. — Vol. 36. — P. 41–57.

[58] Hansen P., Mladenovic N. Variable neighborhood search // Europian J. Oper. Res. — 2001. — V. 130, — P. 449–467.

[59] Hay D.A. Sequential entry and entry-deterring strategies in spatial competition // Oxford Econom. Papers. 1976. V. 28. P. 240 – 257.

[60] Hooker J.N. Integrated Methods for Optimization. — New York: Springer Science+Buisness Media, 2007. — 490 p.

[61] Hooker J.N. Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. — New York: Wiley, 2000. — 520 p.

[62] Hotelling H. Stability in competition // Econom. J. 1929. V. 39. P. 41 – 57.

[63] Kochetov Yu., Alekseeva E., Levanova T. and Loresh M. Large neighborhood local search for the p-median problem // Yugoslav J. Oper. Res. — 2005. — V. 15, — P. 53–63.

[64] Kochetov Yu., Plasunov A. Genetic local search the graph partitioning problem under cardinality constraints // Computational Mathematics and Mathematical Physics. — 2012. — V. 52, — P. 157–167.

[65] Kress D., Pesch E. Sequential competitive location on networks // Eur. J. Oper. Res. 2012. V. 217. P. 483 – 499.

[66] Lodi A., Milano M., Toth P. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems // LNCS. — 2010. — Vol. 6140. — 369 p.

[67] Marriott K., Stuckey P. Programming with Constraints: An Introduction. — Cambridge: The MIT Press, 1998. — 476 p.

[68] McDaniel D., Devine M. A modified Benders partitioning algoritm for mixed integer programming // Management Science. — 1977. — Vol. 24. — P. 312–319.

[69] Mirchandani P.D., Francis R.L. Discrete Location Theory. — New York: Wiley & Sons, 1990. — 555 p.

[70] Noltermeier H., Spoerhose J., Wirth H.C. Muliple voting location and single voting location on trees // Eur. J. Oper. Res. 2007. V. 181. P. 654-667.

[71] Pelegrin B., Fernandez P., Garcia M.D., Cano S. On the location of new facilities for chain epansion under delivered pricing // Omega. 2012. V. 40. P. 149 – 158.

[72] Plastria F. Sequential location problems // Eur. J. Oper. Res. 2001. V. 129. P. 461–470.

[73] Prescott E.C., Vissher M. Sequential location among firms with foresight // Bell J. Econom. Papers. 1977. V. 8. P. 378 – 393.

[74] Rebeiro C.C., Hansen P. Essays and surveys in metaheuristics. — Boston: Kluwer Academic Publishers, 2002. — 651p.

[75] ReVelle, C.S., Eiselt, H.A. Location analysis // European Journal of Operational Research, 2005. Vol. 165. P. 1 – 19.

[76] Serra D. and ReVelle C. Competitive locations and pricing on networks // Geographic Anal. — 1999 — V. 31, — P. 109–129.

[77] Sherali H.D., Soyster A.L. Convergence analysis and algorithmic implications of twodynamic processes toward an oligopoly — competitive fringe equilibrium set // Comput. Oper. Res. 1988. V. 15. P. 69 – 81.

[78] Stackelberg H.V. Marktform und Gleichgewicht. Berlin: Springer, 1934.

[79] Talbi E-G. Metaheuristics: from design to implementation. – Berlin: Wiley, 2009.

[80] van Hentenryck P., Milano M. Hybrid Optimization: : The Ten Years of Cpaior. — New York: Springer Science+Buisness Media, 2011. — 570 p.

[81] Vanderbeck F., Savelsbergh M.W.P. A Generic View of Dantzig-Wolfe Decomposition for Integer Programming // Operations Research Letters. — 2006. — Vol. 34. — P. 296-306.

[82] Vanderbeck F., Wolsey L.A. Reformulation and decomposition of integer programs. — Monreal: Center Operations Research and Econometrics, 2009. — Vol. 2188. — 71 p.

[83] Vives, X. Oligopoly pricing: old ideas and new tools. — Cambridge: MIT Press, 1999.

[84] Yai, F.C. Sequential locations in directional markets // Region. Sci. Urban Econom. 2001. V. 31. P. 535 – 546.

[85] Yates, A.J. Hotelling and the New York stock exchange // Econom. Lett. 1997. V. 56. P. 107 – 110.