РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
ИНСТИТУТ МАТЕМАТИКИ им. С.Л. СОБОЛЕВА

На правах рукописи
УДК 519.21

Кочетов Юрий Андреевич

МЕТОДЫ ЛОКАЛЬНОГО ПОИСКА ДЛЯ ДИСКРЕТНЫХ ЗАДАЧ РАЗМЕЩЕНИЯ

05.13.18 — математическое моделирование, численные методы
и комплексы программ

Диссертация на соискание учёной степени
доктора физико-математических наук

Новосибирск — 2009

Оглавление

Введение **5**

1 Дискретные модели размещения **27**
1.1 Простейшая задача размещения 28
1.2 Многостадийная задача размещения 33
1.3 Задачи размещения с предпочтениями клиентов 37
1.4 Антагонистические размещения 42
1.5 Задачи стандартизации и унификации 46
1.5.1 Задача выбора состава системы технических средств 47
1.5.2 Двухуровневые системы технических средств 49
1.5.3 Динамическая задача с ограничениями на мощности 51
1.5.4 Задача выбора состава системы с частичным внешним финансированием 53

2 Лагранжевы релаксации **58**
2.1 Общая схема 59
2.1.1 Методы субградиентной оптимизации 60
2.1.2 Геометрическая интерпретация 61
2.1.3 Лагранжева декомпозиция 63
2.2 Лагранжевы релаксации для задачи ВСС 64
2.2.1 Сильные и слабые формулировки 64
2.2.2 Нижние оценки оптимума 65
2.2.3 Алгоритм решения задачи $LR_1(\lambda)$ 68
2.2.4 Построение допустимого решения 73
2.2.5 Численные эксперименты 75
2.3 Лагранжевы релаксации для задачи $B2CC$ 76
2.3.1 Две нижние оценки 77
2.3.2 Соотношение величин F и H 81
2.3.3 Результаты тестовых расчетов 82
2.4 Лагранжевы релаксации для задачи ВДСС 83

2.4.1 Нижние оценки 84
2.4.2 Общая схема алгоритма 88
2.4.3 Задачи с ограничениями на номенклатуру изделий 89
2.4.4 Задачи с фактором серийности 90
2.4.5 Результаты тестовых расчетов 92

3 Метаэвристики 94
3.1 Метаэвристики для задач комбинаторной оптимизации 94
3.2 Вероятностные жадные алгоритмы 98
3.2.1 Алгоритм «лидер группы» 100
3.2.2 Условия дополняющей нежесткости 105
3.2.3 Алгоритм «случайный аутсайдер» 107
3.2.4 Принцип ветвления 112
3.2.5 Модификации алгоритмов ЛГ и СА 115
3.3 Вероятностный поиск с запретами 116
3.3.1 Общая схема 117
3.3.2 Асимптотические свойства 119
3.3.3 Варианты алгоритмов 122
3.3.4 Вычислительные эксперименты 124
3.3.5 Направления дальнейших исследований 129
3.4 Генетический локальный поиск 130
3.4.1 Генетический алгоритм 131
3.4.2 Генетический алгоритм для задачи МПК 134
3.4.3 Тестовые примеры 137
3.4.4 Выбор параметров алгоритма 139
3.4.5 Нижние оценки оптимума 140
3.5 Гибридные методы 150
3.5.1 Гибридный генетический локальный поиск 151
3.5.2 Верхние оценки 156
3.5.3 Тестовые расчеты 162

4 Вычислительная сложность локального поиска 164
4.1 Задачи локального поиска 166
4.2 Класс PLS 169
4.3 PLS-полные задачи размещения 176
4.4 Временная сложность локального спуска 183
4.5 Локально седловые точки 189

4.6 Приближенный локальный поиск 195
4.7 Погрешность локальных оптимумов 198

5 Равновесия по Нэшу в игровых моделях размещения 206
5.1 Игровая модель размещения предприятий 207
5.2 Связь с локальными оптимумами 209
5.3 Сложность нахождения равновесных решений 212
5.4 Поиск приближенных равновесий 217
5.5 Сложность алгоритмов локального улучшения 220

6 Электронная библиотека «Дискретные задачи размещения» 222
6.1 Структура библиотеки 223
6.2 Простейшая задача размещения 224
6.2.1 Полиномиально разрешимые примеры 225
6.2.2 Примеры с экспоненциальным числом строгих локальных минимумов 226
6.2.3 Примеры с большим разрывом целочисленности . . 228
6.2.4 Поведение метаэвристик 229
6.3 Примеры с кластеризацией локальных минимумов 237
6.4 Примеры для конкурентной задачи о p-медиане 242

Заключение 247

Литература 249

Введение

Актуальность темы. Объектом исследования настоящей диссертации являются математические модели размещения, типичные для таких областей как стратегическое планирование, стандартизация и унификация технических средств, выбор оптимальной стратегии поведения на конкурентных рынках и др. Предметом исследования являются дискретные оптимизационные задачи, возникающие в рамках таких моделей. Многие из этих задач тесно связаны с задачей минимизации псевдобулевых функций (полиномов от булевых переменных). Этот факт позволяет увидеть связь между моделями, несмотря на их разнородную структуру. Это внутреннее единство моделей, основанное на псевдобулевых функциях, позволяет использовать однотипные методы, а также показывает способы решения новых задач размещения, постоянно возникающих в приложениях. К таким методам, в первую очередь, относятся методы локального поиска и основанные на них так называемые метаэвристики. Именно этому направлению численных методов решения дискретных задач размещения и посвящена настоящая работа.

Выбор данного направления исследований неслучаен. С одной стороны, методы локального поиска легко адаптируются к изменениям математической модели. Их простота и гибкость открывают широкий простор для решения практических задач. С другой стороны, многие принципиальные вопросы остаются открытыми, в частности, вопросы сходимости методов, оценки их погрешности, возможности получения точного решения задачи и доказательства его оптимальности. Многие методы данного класса сконцентрированы на поиске локальных оптимумов задачи, что порождает массу нетривиальных вопросов чисто математического плана, например, вопрос о вычислительной сложности нахождения локального оптимума. Для дискретных задач размещения, как впрочем и для других задач исследования операций, пока не удается подтвердить или опровергнуть гипотезу о существовании полиномиальных алгоритмов нахождения локальных оптимумов. Известно [212], что эти задачи

не могут быть NP-трудными, если NP $\neq$ co-NP, что дает надежду найти такие алгоритмы. Однако эта надежда весьма призрачна в силу плотной PLS-полноты таких задач локального поиска для многих "естественных" окрестностей малой мощности. Доказательство существования полиномиальных алгоритмов было бы слишком сильным результатом. Все задачи из класса PLS решались бы в этом случае за полиномиальное время. Парадокс состоит в том, что на практике методы локального поиска легко находят локальные оптимумы, а метаэвристики, например, генетический локальный поиск, систематически порождая все новые и новые локальные оптимумы, часто предъявляет в качестве ответа глобальный оптимум. Требуются специальные усилия, чтобы найти действительно трудные примеры, где такой подход давал бы заметную ошибку. Однако в теории даже один шаг такого метода может оказаться экспоненциальным по сложности. Теория и практика комбинаторной оптимизации дают здесь принципиально разные оценки возможностей численных методов. Эмпирические исследования свидетельствуют, что для многих NP-трудных задач, в том числе и задач размещения, методы локального поиска позволяют находить приближенные решения, близкие по целевой функции к глобальному оптимуму. Трудоемкость алгоритмов часто оказывается полиномиальной, причем степень полинома достаточно мала.

Теоретические исследования свидетельствуют, что минимальная точная окрестность может иметь экспоненциальную мощность, число шагов для достижения локального оптимума может оказаться экспоненциально большим, значение локального оптимума может сколь угодно сильно отличаться от глобального, а минимальное расстояние между локальными оптимумами может быть сколь угодно большим. Таким образом, актуальность диссертации обусловлена как разработкой и совершенствованием численных методов решения задач размещения, расширением круга решаемых задач и повышением эффективности этих методов, так и исследованием принципиальных возможностей таких методов и, в частности, методов нахождения локальных оптимумов.

Интерес к последней проблеме подогревается целым рядом обстоятельств. Во-первых, стремление получить локальный оптимум оказывается тесно связанным с выделением стационарных точек и условиями Куна–Такера для задач с непрерывными переменными. В идейном смысле понятие окрестности эквивалентно определению вариации в непрерывной оптимизации. Производные в методах комбинаторной оптимизации обычно не используются. Создается впечатление, что классические

методы непрерывной оптимизации слишком далеки от комбинаторных. Тем не менее в ряде случаев условия локальной оптимальности дискретного решения эквивалентны условиям Куна–Такера для непрерывной задачи, полученной релаксацией требования целочисленности переменных. Для NP-трудных задач проблема часто состоит не в том, чтобы найти локальный оптимум, а в том, что таких оптимумов оказывается слишком много. Выделить среди них глобальный оптимум представляется серьезной проблемой, что, по-видимому, и делает исходную задачу NP-трудной. Понятие глобального оптимума не является конструктивным. Даже получив такое решение, доказать его оптимальность очень трудно. Аналогичные проблемы возникают и в непрерывной оптимизации. Там необходимые условия оптимальности также имеют локальный характер и опираются на понятие вариации. В комбинаторной оптимизации реализуется та же идея, и понятие окрестности играет здесь центральную роль.

Вторая причина пристального внимания к локальным оптимумам связана с приближенными алгоритмами, имеющими гарантированные оценки качества. Если комбинаторная задача допускает построение приближенного решения с константной оценкой точности за полиномиальное время, то она по определению принадлежит классу APX [81]. Полиномиальный алгоритм может иметь любую природу и, в частности, может быть не связан с понятием окрестности. Многие комбинаторные задачи не принадлежат этому классу, например, задача о p- медиане, если нет дополнительных предположений о структуре матрицы транспортных затрат. Если же матрица удовлетворяет неравенству треугольника, то задача попадает в класс APX [160]. Оптимизационную задачу P называют *полиномиально ограниченной*, если существует такой полином, что значение целевой функции задачи на любом решении ограничено этим полиномом от длины записи исходных данных. Такие задачи обладают тем свойством, что для любой полиномиально проверяемой окрестности N и любого правила выбора направления спуска стандартный алгоритм локального улучшения является полиномиальным. Для таких задач выделяют класс задач локального поиска, обладающих следующим свойством: для задачи $L = (P, N)$ из класса PLS существует такая константа $\varepsilon > 0$, что каждый локальный оптимум имеет относительную погрешность не более ε. Класс этих задач называют классом GLO (Guaranteed Local Optima). Этот класс не пуст. В нем, например, содержится задача коммивояжера, у которой веса ребер принимают значения 1 или 2. Зада-

ча выполнимости на максимум, задача о максимальном разрезе с единичными весами также принадлежат этому классу [81]. Замыкание класса GLO ($\overline{\text{GLO}}$) определяется с помощью сведения, сохраняющего аппроксимируемость ($\leq_{PTAS}$). Задача P_1 принадлежит замыканию класса GLO, если существует такая задача $P_2 \in$ GLO, что $P_1 \leq_{PTAS} P_2$. Установлено [81], что $\overline{\text{GLO}}$ =APX. Таким образом, класс GLO дает новую характеризацию класса APX, одного из важнейших классов NP-трудных задач. Для любой задачи P из класса APX найдется задача из класса GLO, к которой задача P сводится с сохранением свойств аппроксимируемости. Другими словами, понятие локального оптимума является одним из важнейших в теории аппроксимации NP-трудных задач.

Наконец, последнее, но не менее важное обстоятельство связано с нахождением решений, равновесных по Нэшу, в игровых моделях размещения. В ряде случаев удается установить взаимно-однозначное соответствие между равновесными решениями и локальными оптимумами в специально подобранной оптимизационной задаче. Если такое соответствие обнаружено, то нахождение равновесных решений сводится к уже рассмотренной задаче поиска локальных оптимумов. Более того, если для оптимизационной задачи удается установить ее принадлежность к классу GLO, то тем самым получается оценка для цены анархии в исходной игровой модели. Таким образом проблемы, исследуемые в диссертации, действительно актуальны.

Цель данной работы состоит в разработке и исследовании методов локального поиска для решения дискретных задач размещения, получении апостериорных оценок точности этих методов и построении сложных в вычислительном отношении тестов, позволяющих указать наиболее трудные примеры для этих методов.

Методика исследований. В диссертации используются традиционные методы Лагранжевых релаксаций, динамического программирования, декомпозиции задач путем разложения допустимой области на подмножества. Кроме того, применяются оригинальные итерационные методы, в том числе и точные конечные методы, разработанные автором.

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

1. Разработаны и исследованы вероятностные методы локального по-

иска для ряда дискретных задач размещения. Все рассматриваемые задачи являются NP-трудными. Более того, одна из них, а именно конкурентная задача о p-медиане, является $\sum_2^P$-трудной, то есть имеет более высокий статус сложности, чем любая задача из класса NP. Полученные методы являются итерационными и в асимптотике позволяют находить точное решение задачи. Многочисленные экспериментальные исследования показывают, что фактически оптимум находится достаточно быстро и предлагаемые методы могут с успехом применяться на практике.

2. Для конкурентной задачи о p-медиане разработан точный конечный метод. Предшествующие аналоги основывались на схеме неявного перебора, что существенно ограничивало размерность решаемых задач. Новый метод принципиально отличается от предшествующих. Он основан на оригинальной переформулировке задачи и идеи постепенного наращивания семейства решений Конкурента. Новый метод позволяет существенно увеличить размерность решаемых задач и дает общую схему получения оценок точности для приближенных методов решения данной задачи.

3. Для задач унификации и стандартизации, являющихся обобщением задач размещения, разработаны итерационные методы вычисления апостериорных оценок глобального оптимума. Эти методы основаны на Лагранжевых релаксациях, что требует точного решения релаксированных задач. Исследована сложность получаемых релаксированных задач, получены точные полиномиальные алгоритмы их решения.

4. Исследована вычислительная сложность задачи нахождения локальных оптимумов для дискретных задач размещения с рядом полиномиально проверяемых окрестностей. Показано, что в худшем случае стандартный алгоритм локального улучшения может потребовать экспоненциального числа шагов при любом правиле замещения для каждой из рассматриваемых окрестностей и любого обобщения простейшей задачи размещения или любого обобщения задачи о p-медиане. Установлено, что задачи поиска локального оптимума принадлежат классу PLS (polynomial time local search problems), а некоторые из них являются плотно полными в этом классе. Если же требуется найти не произвольный локальный оптимум, а оптимум, достижимый из заданной стартовой точки, то такая задача часто оказывается PSPACE-полной.

5. Исследована вычислительная сложность нахождения равновесных решений по Нэшу в игровых моделях размещения. Показано, что поиск таких решений в чистых стратегиях является плотно PLS-полной

задачей. В случае приближенных равновесных решений ситуация может коренным образом измениться в зависимости от того, как оценивается допустимая погрешность. Найдены достаточные условия, при которых поиск приближенных равновесных решений по Нэшу является полиномиально разрешимой задачей.

6. Разработана уникальная электронная библиотека тестовых примеров «Дискретные задачи размещения». Она предназначена для проведения экспериментальных исследований и содержит предложенные автором трудные тестовые примеры для различных моделей размещения. Построение и исследование таких тестов дает возможность увидеть границы применимости различных численных методов, проводить сравнительный анализ алгоритмов и проверять разные идеи и гипотезы. Уникальность библиотеки состоит в подборе оригинальных тестов разной вычислительной сложности, имеющих небольшую размерность, но действительно трудных для нахождения точного решения.

Практическая и теоретическая ценность. Работа носит теоретический и экспериментальный характер. Полученные в ней результаты позволяют лучше понять природу задач локального поиска, их сложность и границы возможностей полиномиальных алгоритмов. Разработанные методы реализованы в виде программ. Они показали свою эффективность и могут применяться при решении практических задач, а также при преподавании университетских курсов «Методы оптимизации» и «Теория принятия решений». Электронная библиотека активно используется в научных исследованиях в России и за рубежом для тестирования алгоритмов дискретной оптимизации и сравнительного анализа их эффективности.

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

– Российская конференция «Дискретный анализ и исследование операций», Новосибирск, 1996, 1998, 2000, 2002, 2004;

– Российская конференция «Дискретная оптимизация и исследование операций», Владивосток, 2007;

– Международный симпозиум по исследованию операций (OR), Германия, 1996, 1999, 2000, 2005, 2006, 2008;

– Международная конференция по метаэвристикам (MIC), 2001 (Португалия), 2003 (Япония), 2005 (Австрия);

– Конференция Европейского общества по исследованию операций (EURO), 1998 (Бельгия), 2006 (Исландия), 2007 (Чехия);

– Европейская конференция по методам локального поиска 2005 (Испания);

– Международная конференция «Комбинаторная оптимизация» 1998 (Бельгия);

– Международный симпозиум по математическому программированию (ISMP), 1997(Швейцария);

– Байкальская международная школа-семинар «Методы оптимизации и их приложения», Иркутск, 1995, 2001, 2005, 2008;

– Всероссийская конференция «Математическое программирование и приложения», Екатеринбург, 1999, 2001, 2007;

– Всероссийская конференция «Проблемы оптимизации и экономические приложения», Омск, 1997, 2003, 2006, 2009;

– Конференция европейского общества по задачам размещения (EWGLA), 2000 (Испания);

– Симпозиум по исследованию операций (SYM-OP-IS), 2005 (Черногория).

Результаты диссертации докладывались на семинарах:

– «Математические модели принятия решений», Институт математики им. С.Л. Соболева СО РАН, Новосибирск;

– «Дискретные экстремальные задачи», Институт математики им. С.Л. Соболева СО РАН, Новосибирск;

– «Дискретная оптимизация», Омский филиал Института математики им. С.Л. Соболева СО РАН, Омск;

– «Комбинаторные методы в исследовании операций», Университет Монреаля, Канада;

– Семинар высшей школы коммерции, Монреаль, Канада;

– Семинар национального университета Чар-Тунг, Тайвань.

Публикации. По теме диссертации автором опубликовано 52 работы, в том числе 20 статей, среди них 16 в журналах из списка ВАК, 32 работы в трудах российских и международных конференций.

Основные результаты диссертации

1. Разработаны и исследованы новые вероятностные методы локального поиска для решения NP-трудных задач о p-медиане, простейшей и многостадийной задач размещения, размещения с предпочтениями клиентов, задач стандартизации и унификации, а также для решения более сложной задачи о (r,p)-центроиде (конкурентной задачи о p-медиане), являющейся Σ_2^P–трудной. Эти итерационные методы позволяют находить оптимальные решения и в ряде случаев применяются для доказательства оптимальности получаемых решений.

2. Предложены и обоснованы итерационные методы вычисления апостериорных оценок точности получаемых приближенных решений для указанных выше труднорешаемых задач. В основе этих методов лежат оригинальные точные методы решения релаксированных задач.

3. Установлена вычислительная сложность нахождения локальных оптимумов для ряда задач размещения с полиномиально проверяемыми окрестностями. Показано, что эти задачи принадлежат классу PLS и являются наиболее трудными в этом классе. Получена экспоненциальная нижняя оценка на число итераций для алгоритмов локального улучшения при любом правиле выбора направления спуска. Доказана PSPACE-полнота задачи локального поиска при заданной стартовой точке.

4. Для задач размещения в игровой постановке установлена PLS-полнота задачи нахождения равновесных решений по Нэшу в чистых стратегиях. Получены достаточные условия эффективного вычисления приближенных равновесных решений. Показано, что в общем случае поиск приближенных равновесных решений является столь же сложной задачей, что и поиск самих равновесий по Нэшу.

5) Для экспериментального исследования численных методов и тестирования комплексов программ разработана электронная библиотека «Дискретные задачи размещения». В ней представлены тесты разной вычислительной сложности, оптимальные решения и результаты численных исследований для точных и итерационных методов локального поиска.

Личный вклад. Диссертационная работа представляет единый цикл многолетних исследований автора, объединенных не только предметом,

но и методами исследований. В совместных работах соискателю принадлежат основные идеи новых методов локального поиска, методов построения апостериорных оценок точности, способы исследования вычислительной сложности задач локального поиска и поиска равновесных решений, а также идеи конструкций сложных тестовых примеров. Отдельные элементы доказательств утверждений и теорем выполнены в соавторстве при непосредственном участии соискателя. Конфликт интересов с соавторами отсутствует.

На защиту выносятся совокупность результатов по разработке и исследованию методов локального поиска для дискретных задач размещения, методы вычисления апостериорных оценок точности, результаты по вычислительной сложности задач локального поиска и получению равновесных решений по Нэшу, а также конструкции сложных тестовых примеров.

Объем и структура диссертации. Диссертация состоит из введения, шести глав и списка литературы (212 наименований). Объем диссертации — 267 страниц.

Содержание работы

Интерес к задачам размещения связан в первую очередь с приложениями, что характерно для задач исследования операций. Исторически, первые задачи этого класса стали исследоваться в Институте математики им. С.Л. Соболева СО РАН с конца 60-х годов 20-го столетия. Математические модели формулировались в терминах состава систем технических средств и формально не были связаны с размещением [10]. Позже, когда связь этих двух направлений стала проявляться все более и более отчетливо, центр тяжести исследований постепенно стал перемещаться в сторону задач размещения. Все рассматриваемые задачи оказывались NP-трудными. Это обстоятельство требовало сконцентрировать внимание на принципиальных вопросах разработки численных методов, найти эффективные подходы к решению задач данного класса, выделить наиболее трудные тестовые примеры.

Первая глава диссертации дает краткий обзор моделей размещения и тесно связанных с ними моделей стандартизации и унификации, ко-

торыми занимался автор с 1979 года. В первую очередь это простейшая задача размещения или, следуя [10], задача выбора оптимального ряда изделий. Этой задаче посвящены десятки статей, главы в монографиях. Хорошо известна связь задачи с псевдобулевыми функциями (полиномами от булевых переменных). Этот вопрос подробно исследуется в книге В. Береснева [9]. Однако до сих пор эквивалентность простейшей задачи размещения специальному классу псевдобулевых функций использовалась только в теории. В [8] отмечено, что одной псевдобулевой функции могут соответствовать разные примеры простейшей задачи размещения. Бесконечное число примеров может порождать одну и ту же псевдобулеву функцию, причем размерность примеров может сильно варьироваться. Так как время решения задачи зависит от размерности, то можно перед решением задачи постараться максимально сократить эту размерность, используя псевдобулеву функцию и заменяя пример на эквивалентный, но с минимальной размерностью. Эта идея лежит в основе следующего утверждения.

Теорема 1. *Для задачи минимизации псевдобулевой функции с положительными коэффициентами при нелинейных членах эквивалентная ей простейшая задача размещения с минимальным числом клиентов может быть найдена с полиномиальной трудоемкостью от n и m, где m — число переменных псевдобулевых функций, а n — число ее нелинейных членов.*

Аналогичные утверждения доказаны и для многостадийной задачи размещения, а также для задачи размещения с предпочтениями клиентов, когда соответствующая псевдобулева функция имеет имеет слагаемые произвольного знака.

Значительно более важными с практической точки зрения являются конкурентные задачи размещения, когда несколько лиц, принимающих решение (игроков), стремятся разместить свои предприятия и разделить рынок (множество клиентов) для максимизации собственной прибыли. В диссертации рассматриваются два вида таких игр: игроки действуют последовательно или одновременно. Последнему случаю посвящена пятая глава диссертации. В первой главе приводится постановка задачи при последовательном принятии решений двумя игроками: Лидером и Конкурентом. Показана связь такой игры Штаккельберга с псевдобулевыми функциями. Последний раздел первой главы посвящен задачам стандартизации и унификации. Приводятся постановки задач выбора ря-

да изделий при многоэтапном процессе выполнения работ, когда часть работ области применения системы выполняется на первом этапе, часть — на втором и т.д. Начальный состав системы и объемы возможного пополнения считаются известными. Задача состоит в выборе такого состава системы, который позволяет выполнить все работы с минимальными суммарными затратами. Простейшая задача размещения и классическая задача о p-медиане являются частными случаями этой задачи. Кроме того, приводится постановка динамической задачи выбора ряда изделий, а в последнем разделе — постановка задачи выбора ряда изделий с частичным внешним финансированием (ВР). Показано, что эта задача может быть сведена к серии простейших задач размещения.

Теорема 4. *Решение задачи ВР сводится к решению m задач выбора ряда изделий, m — число изделий исходного ряда.*

Этот результат получен совместно с А.В. Плясуновым.

Вторая глава диссертации посвящена методам вычисления апостериорных оценок оптимума в задачах размещения и стандартизации. Идея получения таких оценок основана на Лагранжевых релаксациях. Часть ограничений задачи заносится в целевую функцию с множителями Лагранжа. К полученной двойственной задаче применяются известные методы субградиентной оптимизации. Каждый раз решение релаксированной задачи представляет определенную трудность, связанную как с выбором наиболее сильной формулировки в терминах целочисленного линейного программирования, так и с получением точного решения при фиксированных множителях Лагранжа. Такое решение позволяет вычислить субградиент оптимизируемой функции и применить схемы негладкой оптимизации. Результаты данного подхода иллюстрируются численными экспериментами, подтверждающими работоспособность метода и возможности построения приближенных решений с малой относительной погрешностью.

В первом разделе приводится общая схема Лагранжевых релаксаций для задач целочисленного линейного программирования. Второй раздел посвящен верхним и нижним оценкам оптимума в задаче выбора оптимального состава системы при многоэтапном процессе выполнения работ. Рассматриваются две Лагранжевы релаксации задачи. Установлены соотношения между оптимумами в соответствующих двойственных задачах и оптимумами в линейных релаксациях, получающихся заменой

условий булевости переменных на условие их неотрицательности. Показано, что одна из релаксированных задач разрешима за время, полиномиальное от размерности задачи. Этот результат позволяет построить итерационный процесс, на каждом шаге которого за полиномиальное время получается точное решение исходной задачи с измененными правыми частями релаксированного ограничения. Такое решение достраивается до допустимого решения исходной задачи с помощью жадной эвристики. Предложенный способ получения верхних оценок оптимума страдает рядом недостатков. В частности, он ориентируется на оптимум двойственной задачи, который при наличии разрыва целочисленности может сильно отличаться от оптимума в исходной задаче. В связи с этим в конце второго раздела исследуются модификации предложенной схемы, в том числе и комбинированный метод, использующий обе релаксации и сглаживающий их недостатки.

В третьем разделе рассматривается обобщение предшествующей модели, в которой технические средства могут иметь унифицированные составные части. Многостадийные задачи размещения являются частным случаем этой задачи. Для получения верхних и нижних оценок оптимума рассматриваются две Лагранжевы релаксации задачи по разным группам ограничений. При фиксированных множителях Лагранжа решения этих задач дают нижние оценки оптимума. Поиск наилучших множителей приводит к двойственным задачам D_f и D_h. Установлено, что эти задачи решаются с полиномиальной трудоемкостью.

Теорема 7. *Задачи D_f и D_h полиномиально разрешимы.*

Практическое использование этих оценок зависит от разрыва целочисленности, который может быть различным для этих двойственных задач. Показано, что полученные нижние оценки несравнимы между собой. Более точно, приводится семейство примеров исходных данных, когда одна оценка доминирует другую. В ходе численных экспериментов исследуются области доминирования одной оценки над другой в зависимости от жесткости ограничений на объемы производства изделий и начальный состав системы. Получены практические рекомендации по применению оценок D_f и D_h.

В последнем разделе второй главы исследуются возможности Лагранжевых релаксаций для задачи выбора динамического состава системы (задача ВДСС). Показано, что решение релаксированной задачи сводится к решению серии задач линейного программирования. Выделяются

частные случаи, когда двойственная задача также полиномиально разрешима. Приводится семейство исходных данных, на котором абсолютный разрыв целочисленности может достигать произвольно больших размеров. Для построения приближенного решения задачи разработана итерационная процедура, существенно опирающаяся на решение двойственной задачи. В конце раздела рассматриваются модели с ограничениями на номенклатуру изделий. Выделяются два вида таких моделей. В первом случае задается ограничение на обобщенный ресурс, выделяемый на весь интервал планирования. Во втором случае этот ресурс делится по годам и не передается из одного года в другой. В обоих случаях снова применяется идея Лагранжевых релаксаций, которая приводит к многовариантной задаче о рюкзаке или обобщенной задаче о назначениях, что позволяет использовать уже известные подходы.

Стоимость производства изделий предполагается постоянной величиной, что существенно упрощает модель ВДСС. В последнем разделе этой главы рассматривается более общая ситуация, когда стоимость производства изделий зависит от суммарного объема выпуска изделий. Более того, для каждого изделия исходного ряда заданы интервалы серийности. Внутри каждого интервала удельная стоимость производства остается постоянной величиной, а при переходе от одного интервала в другой может меняться произвольно. Алгоритм решения такой задачи также основан на полиномиальной разрешимости релаксированной задачи. В ходе численных экспериментов исследуется зависимость погрешности получаемых решений от величины начальных затрат на ввод в систему новых образцов. При больших значениях этого параметра исходная задача вырождается и фактически становится задачей о минимальном покрытии множества. Начальные затраты доминируют в целевой функции, и если алгоритму удается найти минимальную номенклатуру для выполнения всех работ, то получается точное решения задачи. В противном случае относительное уклонение от оптимума достигало 10%.

Результаты этой главы получены совместно с М.Г. Пащенко.

Третья глава является самой объемной. В ней приводится описание метаэвристик и результаты их экспериментальных исследований для дискретных задач размещения. Первый раздел этой главы носит вводный характер. Он содержит классификацию данных методов, основные идеи метаэвристик, обзор литературы и пути гибридизации с классическими методами математического программирования. Второй раздел

посвящен конструктивным эвристикам. В нем исследуются два вероятностных жадных алгоритма на примере многостадийной задачи размещения. Первый алгоритм, *Лидер группы*, является рандомизированной версией хорошо известного алгоритма координатного спуска. Введение рандомизации позволяет существенно сократить погрешность получаемых решений, повысить частоту нахождения точного решения задачи и решений с заданной погрешностью.

Второй алгоритм, *Случайный аутсайдер*, опирается на условие дополняющей нежесткости. Он предполагает последовательное сокращение числа открытых предприятий. Закрываемое предприятие выбирается случайным образом из множества наименее привлекательных предприятий. Эти алгоритмы оказались несравнимыми между собой. На одних тестах выигрывает один подход, на других тестах — другой. Заметное сокращение погрешности и повышение частоты нахождения точного решения задачи получено на пути гибридизации этих алгоритмов с усеченным методом ветвей и границ. Десяти ветвлений оказалось достаточно, чтобы сгладить различия в погрешности алгоритмов и поднять частоту нахождения оптимума.

Результаты этого раздела получены совместно с Е.Н. Гончаровым.

Третий раздел посвящен вероятностному методу поиска с запретами. Излагается общая схема метода, предложенная Ф. Гловером [129], и ее новый рандомизированный вариант. Показано, что вероятностный алгоритм поиска с запретами при определенных ограничениях на структуру и длину списка запретов порождает неразложимую цепь Маркова. Как следствие получается, что при любом стартовом решении вероятность получения точного решения задачи стремится к единице с ростом числа шагов алгоритма. Так как нетривиальных оценок на скорость сходимости получить не удается, а в силу NP-трудности рассматриваемых задач, по-видимому, и не удастся, то наибольший интерес представляют экспериментальные исследования поведения таких методов на различных классах тестовых примеров. Проведенные исследования свидетельствуют о высокой частоте получения точного решения даже на трудных в вычислительном отношении тестах. Показано положительное влияние рандомизации окрестности на результаты работы алгоритма, исследованы различные стратегии диверсификации поиска, получены доверительные интервалы на вероятность получения точного решения задачи при заданном числе итераций и приближенного решения с погрешностью не более одного процента.

Четвертый раздел посвящен генетическому локальному поиску. Приводятся общая схема этого популярного в комбинаторной оптимизации итерационного метода, обсуждение ключевых элементов данной схемы, асимптотических свойств и интерпретация в терминах цепей Маркова. Эффективность данного метода во многом зависит от выбора применяемых операций скрещивания, мутаций и процедур локального улучшения, то есть от того, насколько удачно адаптированы эти ключевые элементы под специфику решаемой задачи. В данном разделе исследуются возможности генетического локального поиска для решения задачи о p-медиане с предпочтениями клиентов (МПК). Рассматриваются четыре оператора скрещивания и две окрестности: стандартная окрестность $Swap$ и новая окрестность для задач размещения, построенная на идеях Лина и Кернигана [157]. Исследуется влияние разных стратегий селекции и мутаций на поведение алгоритма. Для получения нижних оценок оптимума рассмотрены различные возможности сведения задачи к целочисленному линейному программированию. Получено новое сведение, основанное на известной задаче о паре матриц. Установлено, что новое сведение может давать лучшую нижнюю оценку (обозначенную в работе через LB_6), чем предшествующие известные оценки. Лучшая из них обозначена через LB_4, причем $LB_6 \geq LB_4$.

Теорема 12. *Для любого $N > 0$ существуют такие исходные данные задачи МПК, что $LB_6/LB_4 \geq N$.*

Предложена конструкция таких исходных данных. Результаты этого раздела получены совместно с Е.В. Алексеевой.

Последний раздел третьей главы посвящен гибридным метаэвристикам. Обсуждаются различные возможности гибридизации, исследуется одна из таких возможностей: гибридная схема генетического алгоритма и вероятностного поиска с запретами на примере конкурентной задачи о p-медиане. Приводится общая схема такого подхода, обсуждаются ее основные элементы и проблемы реализации. Основной проблемой является процедура локального улучшения. Так как поиск допустимого решения задачи связан с решением NP-трудной задачи Конкурента, то нахождение лучшего элемента в окрестности требует многократного обращения к решению этой задачи. В итоге просмотр окрестности превращается в чрезвычайно сложную задачу. Для решения этой проблемы используется переход к линейной релаксации в задаче Конкурента и рандомизация окрестности. Эти два приема резко сокращают трудоемкость просмотра

окрестности, делая процедуру полиномиальной. Исследуются результаты столь кардинального изменения сложности локального поиска. Для получения верхних оценок исходной $\sum_2^P$–трудной задачи предлагаются два подхода. Первый из них связан с введением дополнительных ограничений в задачу Конкурента. Второй способ основан на сужении области поиска, когда исходная задача переписывается в виде задачи линейного частично-целочисленного программирования с экспоненциальным числом переменных и ограничений, а затем оставляется небольшое семейство этих переменных и ограничений. Оптимальное решение в такой задаче дает верхнюю оценку на доход Лидера. Систематически пополняя семейство, можно получить точное решение задачи. Возможности такого подхода исследуются в последней главе диссертации.

Результаты этого раздела получены совместно с А.В. Плясуновым и Е.В. Алексеевой.

В четвертой главе изучаются вопросы вычислительной сложности задачи нахождения локального оптимума. В первом разделе вводится понятие задачи локального поиска и даются примеры задач, когда такой подход приводит к точному решению. Во втором разделе дается краткое введение в теорию сложности задач локального поиска. Вводится класс PLS, определение сведений в этом классе и список PLS-полных задач. В третьем разделе устанавливается вычислительная сложность локального поиска для различных задач размещения.

Теорема 19. *Задача поиска локального минимума для простейшей задачи размещения с окрестностью Flip является PLS-полной.*

Для классической задачи о p-медиане получено достаточное условие, при котором соответствующая задача локального поиска будет PLS-полной.

Следствие 5. *Для полноты задачи (p-медиана, N) из класса PLS достаточно, чтобы окрестность N была не слабее окрестности FM_1.*

В четвертом разделе исследуется временная сложность стандартного алгоритма локального улучшения. Вводится понятие плотной PLS-полноты и доказывается, что задача о p-медиане с рядом полиномиально проверяемых окрестностей является плотно PLS-полной задачей.

Теорема 23. *Задача о p-медиане с каждой из окрестностей Swap, LK, LK_1, FM, FM_1 является плотно PLS-полной.*

Аналогичное свойство установлено и для простейшей задачи размещения с окрестностью $Flip$. Полученные свойства приводят к экспоненциальным нижним оценкам для числа шагов стандартного алгоритма локального улучшения в худшем случае. Если же требуется найти локальный минимум, достижимый из заданной точки локальными улучшениями по указанным окрестностям, то такая задача становится PSPACE-полной.

Теорема 25. *Для простейшей задачи размещения с окрестностью Flip и задачи о p-медиане с окрестностями Swap, LK, LK_1, FM, FM_1 соответствующие стандартные задачи локального поиска являются PSPACE-полными.*

В пятом разделе исследуется связь задачи нахождения локальных оптимумов с классическими условиями Куна-Такера. Для задачи минимизации произвольного полинома от булевых переменных на заданном слое гиперкуба вводится функция Лагранжа и понятие седловой точки относительно окрестности $Swap$. Установлено, что точка гиперкуба будет локальным минимумом относительно окрестности $Swap$ тогда и только тогда, когда соответствующее ей решение удовлетворяет условиям Куна-Такера.

В шестом разделе исследуются возможности получения приближенных локальных оптимумов. Допустимое решение s^ε задачи (OP, N) называют ε-локальным минимумом, если $(F(s^\varepsilon) - F(s))/F(s) \leq \varepsilon$ для всех $s \in N(s^\varepsilon)$, F —целевая функция задачи OP. Другими словами, в окрестности $N(s^\varepsilon)$ могут содержаться решения с меньшим значением целевой функции, но их относительное уклонение от s^ε не должно превышать ε. Показано, что для дискретных задач размещения из класса PLS, имеющих линейную целевую функцию, существует полностью полиномиальная ε-локальная оптимизационная схема, то есть семейство алгоритмов $(A_\varepsilon)_{\varepsilon>0}$, позволяющее находить ε-локальный минимум для любого $\varepsilon > 0$ за время, ограниченное полиномом от длины записи исходных данных и величины $1/\varepsilon$.

В последнем разделе изучаются возможности получения локальных оптимумов с гарантированной оценкой точности и принципиальная возможность построения точных окрестностей. *Точной окрестностью* называют такую окрестность, для которой любой локальный оптимум является глобальным. Показано, что построение таких окрестностей сопряжено с принципиальными трудностями.

Теорема 29. *Если P≠NP, то для любого $\rho > 1$ и любой полиномиально проверяемой окрестности найдутся исходные данные задачи о p-медиане, для которых существует локальный минимум, отклоняющийся более чем в ρ раз от оптимального значения задачи.*

При $\rho = 1$ получаем следующее утверждение.

Следствие 9. *Если P≠NP, то для задачи о p-медиане не существует точных полиномиально проверяемых окрестностей.*

Оптимизационную задачу OP называют *псевдо-полиномиально ограниченной*, если существует такой полином p от переменных $|x|$ и $Max(x)$ — наибольшее по величине число из входа x, что для любого входа x и любого допустимого решения s выполняется неравенство

$$F(s,x) - Opt(x) \leq p(|x|, Max(x)),$$

где $Opt(x)$ — глобальный минимум для входа x. Множество таких оптимизационных задач обозначают NPO_{PPB}.

Теорема 30. *Пусть $OP \in NPO_{PPB}$, $(OP, N) \in PLS$ и целевая функция принимает только целочисленные значения. Если P≠NP и задача вычисления приближенного решения с гарантированной оценкой точности ρ является NP-трудной в сильном смысле, то найдутся исходные данные, для которых существует локальный оптимум, отклоняющийся более чем в ρ раз от оптимального значения.*

Следствие 10. *Если P≠NP и задача $OP \in NPO_{PPB}$ является NP-трудной в сильном смысле, то для нее не существует точных полиномиально проверяемых окрестностей.*

В заключение главы приводится семейство исходных данных задачи о p-медиане, для которого оптимальное решение задачи отстоит от любого локального минимума на максимально возможном расстоянии, то есть на расстоянии, равном диаметру допустимой области. Эти локальные оптимумы образуют своего рода "плато", отделенное от глобального оптимума большими значениями целевой функции. Погрешность таких локальных оптимумов может оказаться сколь угодно большой величиной.

Результаты этой главы получены совместно с А.В. Плясуновым.

Пятая глава посвящена игровым моделям размещения. В отличие от игр Штаккельберга, рассматривающихся в первой и третьей главах,

здесь исследуется ситуация равноправных игроков. Они не образуют коалиций и преследуют чисто эгоистические цели: максимизация собственной прибыли от открытия предприятий и обслуживания клиентов. Игроки принимают решения одновременно. Тот факт, что они не образуют коалиций и имеют полную информацию о поведении друг друга, приводит к падению цен на рынке и перераспределению доходов между игроками и потребителями. Решение называют равновесным по Нэшу, если ни один из игроков не может увеличить свою прибыль при условии, что другие игроки не меняют свои решения. Понятие равновесного решения в чистых стратегиях оказывается тесно связанным с понятием локального оптимума в соответствующей оптимизационной задаче. Таким образом удается установить сложностной статус задачи поиска равновесных решений. Показано, что задача поиска равновесия принадлежит классу PLS и является плотно полной в нем. Отсюда, в частности, следует, что доказательство существования полиномиального алгоритма вычисления равновесного решения было бы слишком сильным результатом. Все задачи из класса PLS решались бы в этом случае за полиномиальное время. Доказательство обратного утверждения приводит к выводу, что P$\neq$NP. Таким образом, вопрос о сложности нахождения равновесных решений в чистых стратегиях является интригующим и может иметь грандиозные последствия.

В первом разделе приводится формальное описание игровой модели. Во втором разделе установается взаимно-однозначное соответствие между равновесными по Нэшу решениями и локальными максимумами некоторой специально сконструированной задачи FLG (Facility Location Game). В качестве окрестности N_1 текущего решения рассматриваются все решения, получаемые выбором игрока и заменой его решения на любое другое допустимое решение. В третьем разделе устанавливается плотная PLS полнота задачи нахождения равновесного решения.

Теорема 33. *Задача* $(Max - Cut, Flip)$ *плотно PLS–сводится к задаче* (FLG, N_1).

Показано, что утверждение остается верным и для случаев, когда стоимости открытия предприятий равны нулю, и когда каждый игрок может открывать несколько предприятий.

В четвертом разделе исследуется сложность получения приближенных равновесных решений. Если игрок k меняет свою стратегию, он несет расходы в размере Δ_k. Игрок не будет менять свое решение, ес-

ли приращение прибыли не превосходит данной величины. Если в такой ситуации ни один из игроков не может увеличить свою прибыль при условии, что другие игроки не меняют свой выбор, то такое решение называют *приближенным равновесным решением* с погрешностью $\Delta = (\Delta_1, \ldots, \Delta_p)$, где p — число игроков. Задача поиска приближенных равновесий кажется простой, так как увеличивается число подходящих состояний. Тем не менее это не всегда так. Если Δ_k являются постоянными величинами, то поиск приближенных равновесий снова оказывается плотно PLS-полной задачей. Если же для любых стратегий $(i_1, \ldots, i_p)$ величины Δ_k могут быть оценены снизу функционалом $\mu(i_1, \ldots, i_p)$ суммарной прибыли игроков и экономии клиентов (социальное благо от размещения предприятий и выпуска продукции), то есть $\Delta_k \geq \varepsilon\mu(i_1, \ldots, i_p)$ для любого $k = 1, \ldots, p$, то задача может быть решена с полиномиальной трудоемкостью.

Теорема 35. *Если $\Delta_k \geq \varepsilon\mu(i_1, \ldots, i_p)$ для любого $k = 1, \ldots, p$ и любых стратегий $(i_1, \ldots, i_p)$, то приближенное равновесие с погрешностью $\Delta = (\Delta_1, \ldots, \Delta_p)$ может быть найдено за полиномиальное время от длины записи исходных данных и величины $1/\varepsilon$.*

Теорема 36. *Если Δ_k являются постоянными величинами, то поиск приближенного равновесия является плотно PLS–полной задачей.*

В последнем разделе исследуется сложность нахождения равновесных решений, достижимых из заданного начального состояния.

Теорема 38. *Нахождение равновесного решения при заданной начальной позиции игроков является PSPACE-полной задачей.*

Пусть некоторое решение не является равновесным, то есть существует игрок, возможно не один, который может увеличить свою прибыль, выбрав другие предприятия. Выбор такого игрока и замену его предприятий назовем одним шагом итерационной процедуры улучшения. Процесс выбора игрока и поиск для него лучших предприятий могут иметь произвольную вычислительную сложность, быть детерминированными или рандомизированными алгоритмами. Показано, что в худшем случае такая итерационная процедура требует экспоненциального числа шагов для нахождения равновесного решения.

Результаты этой главы получены совместно с А.В. Плясуновым.

Шестая глава диссертации посвящена электронной библиотеке тестовых примеров «Дискретные задачи размещения».

В первом разделе приводится общая структура библиотеки. Во втором разделе дается описание трудных в вычислительном отношении тестовых примеров для простейшей задачи размещения. Здесь представлены восемь классов тестовых примеров, сильно отличающиеся по трудоемкости нахождения точного решения:

– примеры, использующие конструкцию конечных проективных плоскостей;

– примеры, использующие совершенные коды с расстоянием 3;

– примеры, базирующиеся на шахматных торах;

– три класса примеров с большим разрывом целочисленности (классы GapA, GapB, GapC);

– примеры с матрицами расстояний на двумерной евклидовой плоскости;

– примеры с матрицей расстояний, элементы которой порождаются случайным образом с равномерным распределением из заданного интервала.

Первый класс является полиномиально разрешимым. Зная способ порождения примеров, легко найти точное решение задачи. Однако для методов локального поиска он оказывается достаточно трудным [185, 186], так как каждому пучку прямых соответствует локальный минимум задачи с большим бассейном притяжения.

Второй и третий классы интересны тем, что число строгих локальных оптимумов для них растет экспоненциально с ростом размерности задачи. Так как по построению любой из них может оказаться глобальным оптимумом, то процесс решения задачи может быть достаточно долгим для любого метода, в том числе и для метаэвристик.

Три класса с большим разрывом двойственности являются сложными для метода ветвей и границ [158]. Разрыв составляет более 20%, что приводит к огромному дереву ветвлений с несколькими десятками миллионов вершин даже при небольшой размерности задачи, $n = m = 100$. Самый сложный из этих классов, *Gap–C* является также трудным и для методов локального поиска: вероятностного поиска с запретами, генетического алгоритма и схемы GRASP [105, 179]. Частота нахождения оптимального решения не превышает 0,53 при выполнении этими алгоритмами 10^4 итераций, соответствующих переходу от текущего решения к соседнему для объединения окрестностей Flip и Swap. Последние два клас-

са являются наиболее легкими. При заданной размерности $n = m = 100$ такие примеры легко решаются как методом ветвей и границ, так и метаэвристиками.

В третьем разделе приводится семейство исходных данных многостадийной задачи размещения, для которых локальные оптимумы группируются в три кластера. Один кластер характеризуется большими значениями целевой функции, два других — малыми значениями. В задаче имеется один глобальный минимум. Чтобы попасть к нему локальными изменениями решений из другого кластера, требуется сначала посетить кластер с большими значениями целевой функции. Построенное семейство исходных данных является своего рода "ловушкой" для таких метаэвристик, как вероятностный поиск с запретами, генетические алгоритмы, локальный поиск с чередующимися окрестностями и др., если в алгоритмах не предусмотрены специальные процедуры диверсификации поиска.

Результаты этого раздела получены совместно с Е.Н. Гончаровым.

В последнем разделе обсуждаются примеры для конкурентной задачи о p-медиане и поведение точного метода генерации переменных и ограничений. Приводятся зависимости числа итераций метода при изменении числа открываемых предприятий Лидером и Конкурентом. Обсуждаются пути повышения эффективности предложенного подхода.

Глава 1

Дискретные модели размещения

Модели размещения составляют широкий пласт математических моделей исследования операций, интересный как с практической точки зрения, так и с точки зрения теории комбинаторной оптимизации. Своими корнями это направление уходит к П. Ферма (1601-1665) и Е. Торричелли (1608-1647) [108], но как самостоятельное направление оно сформировалось в 70-80 годы прошлого столетия. На сегодняшний день имеется ряд монографий в этой области [9, 10, 106, 111, 169]. Ежегодно проводятся специализированные конференции, организуемые европейской рабочей группой EWGLA (http://www.vub.ac.be/EWGLA/) и американской рабочей группой SOLA (http://www.ent.ohiou.edu/∼thale/sola/sola.html).

В СССР пионерами этого направления были В. Черенин, В. Хачатуров, В. Трубин, С. Лебедев, а в Сибирском отделении Академии наук В. Береснев, Э. Гимади, В. Дементьев. Столь большой интерес к данной проблематике связан в первую очередь с приложениями, которые возникают не только при размещении предприятий, складов и магазинов, планировании метро, размещении узлов сетей связи и др. Многочисленные приложения возникают, в частности, при решении задач унификации и стандартизации, когда выбирается состав системы технических средств, предназначенных для выполнения заданного списка работ [9]. В качестве целевой функции используются либо величина суммарных затрат на создание и функционирование системы технических средств, либо суммарная эффективность системы, т.е. объем выполняемых работ.

В данной главе рассматриваются дискретные модели размещения, обсуждается их связь с задачами минимизации псевдобулевых функций и задачами стандартизации и унификации. В первом разделе формулируется простейшая задача размещения, обсуждаются различные ее формулировки в терминах целочисленного линейного программирова-

ния (ЦЛП) и в терминах полиномов от булевых переменных. Показано, как с помощью псевдобулевых функций можно найти эквивалентную формулировку задачи с минимальной размерностью. Во втором и третьем разделах рассматриваются многостадийная задача размещения и задача размещения с предпочтениями клиентов. Известно, что обе эти задачи также могут быть эквивалентным образом переформулированы в терминах псевдобулевых функций. По аналогии с первым разделом для этих задач также показана возможность нахождения эквивалентных формулировок с минимальной размерностью. В четвертом разделе формулируется конкурентная задача размещения, в которой два лица последовательно принимают решения об открытии предприятий. Приводится математическая формулировка задачи в терминах псевдобулевых функций и показана ее связь с минимаксными и максиминными задачами. В последнем разделе рассматриваются математические модели стандартизации и унификации. Эти модели тесно связаны с дискретными моделями размещения, но имеют свою логику развития. Приводятся четыре модели: многоэтапная, двухуровневая, динамическая и модель с частичным внешним финансированием. Все они оказываются тесно связанными с простейшей задачей размещения и допускают те же методы решения.

1.1 Простейшая задача размещения

В подавляющем большинстве рассматриваемых моделей размещения авторы исходят из предположения, что имеется одно лицо, принимающее решение (ЛПР), например, руководитель фирмы, размещающий свои предприятия. Для заданного множества клиентов $J = \{1, \ldots, n\}$ он знает производственно–транспортные расходы $c_{ij} \geq 0$, связанные с производством и доставкой продукции j-му клиенту из i-го пункта производства, если оно будет там открыто. Множество возможных пунктов производства $I = \{1, \ldots, m\}$ предполагается конечным, и для каждого пункта $i \in I$ известна стоимость $f_i \geq 0$ открытия предприятия в этом пункте. Задача состоит в выборе такого подмножества $S \subseteq I$ пунктов размещения производства, которое позволяет обслужить всех клиентов с минимальными суммарными затратами, т.е. найти

$$\min_{S \subseteq I} \Big\{ \sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} c_{ij} \Big\}.$$

Первое слагаемое в целевой функции задает затраты на открытие предприятий, второе слагаемое определяет производственно–транспортные расходы. Снабжение клиентов производится из предприятий, которые обеспечивают минимальные производственно–транспортные расходы.

Сформулированная задача известна в литературе как простейшая задача размещения (ПЗР). Она является NP–трудной в сильном смысле даже в случае, если матрица (c_{ij}) удовлетворяет неравенству треугольника [160]. Для построения приближенных алгоритмов эта задача также является сложной. Так как задача о покрытии является её частным случаем, то существование полиномиального приближенного алгоритма, относительная погрешность которого растет медленнее $\log m$, влечет совпадение классов P и NP. Если матрица (c_{ij}) удовлетворяет неравенству треугольника, то лучший на сегодняшний день приближенный полиномиальный алгоритм имеет оценку относительной погрешности 0.52, но и в этом случае существование приближенного полиномиального алгоритма с оценкой меньше 0.463 влечет P=NP. Если же предприятия и клиенты задаются точками в k-мерном евклидовом пространстве, а матрица (c_{ij}) определяет расстояния между ними, то для любого $\varepsilon > 0$ существует приближенный алгоритм с относительной погрешностью не более ε и трудоемкостью, полиномиально зависящей от n, m и экспоненциально зависящей от k и $1/\varepsilon$.

Для точного решения задачи разрабатывались методы ветвей и границ с нижней оценкой, получаемой линейным программированием. Такие оценки зависят от математической формулировки задачи. Первые попытки в этом направлении были неудачными. Они основывались на так называемой "слабой" формулировке задачи.

Введем переменные:

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

$$x_{ij} = \begin{cases} 1, & \text{если клиент } j \text{ снабжается из предприятия } i, \\ 0 & \text{в противном случае.} \end{cases}$$

Тогда простейшая задача размещения может быть записана в виде задачи целочисленного линейного программирования: найти

$$\min \Big\{ \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij} \Big\}$$

при условиях:

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

$$nx_i \geq \sum_{j \in J} x_{ij}, \quad i \in I,$$

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

Замена условий булевости переменных на условия $x_i, x_{ij} \in [0, 1]$, $i \in I, j \in J$ приводит к равенству

$$x_i = \frac{1}{n} \sum_{j \in J} x_{ij}, i \in I$$

и, как следствие, к новой задаче линейного программирования:

$$\min \sum_{j \in J} \sum_{i \in I} (f_i/n + c_{ij}) x_{ij}$$

при условиях:

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

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

Эта задача легко решается точно и дает нижнюю оценку для простейшей задачи размещения. Нижняя оценка получается за полиномиальное время, но является грубой. Замена условия

$$nx_i \geq \sum_{j \in J} x_{ij}, \quad i \in I,$$

на эквивалентное с точки зрения целочисленного оптимума

$$x_i \geq x_{ij}, \quad i \in I, j \in J,$$

приводит к новой линейной программе, которая уже не решается аналитически, но гарантирует более точную нижнюю границу. Именно эта *сильная* формулировка использовалась многими авторами для разработки более совершенных точных методов. Независимо друг от друга В. Береснев [6] в Новосибирске, В. Трубин [60] в Киеве, С. Лебедев [47] в Москве, Я. Краруп, О. Билд [92] в Амстердаме и Д. Эрленкоттер [112] в США использовали эту идею и получили примерно одинаковые результаты численных экспериментов. Разработанный метод легко решает задачи

при $n, m \leq 100$. Для задач большей размерности разработаны специализированные алгоритмы [82, 83, 143], основанные на той же сильной формулировке. Современный взгляд на проблему выбора формулировки задачи можно найти в [194] (см. также [106]).

Наряду с простейшей задачей размещения широкую известность приобрела задача о p–медиане [106]. В этом варианте задачи размещения предполагается, что $f_i = 0, i \in I$, но открывать можно не более p предприятий, $|S| \leq p$. Математическая постановка задачи имеет вид: найти

$$\min_{S \subset I, |S| \leq p} \sum_{j \in J} \min_{i \in S} c_{ij}.$$

В [163] исследуется взаимосвязь этой задачи с простейшей задачей размещения и задачей о покрытии множествами. Существует тесная связь между простейшей задачей размещения и задачей минимизации псевдобулевых функций [9]. Впервые это было отмечено в работах П. Хаммера и С. Рудеану [141]. Позже в работах В. Береснева [8] было предложено новое оригинальное сведение простейшей задачи размещения к задаче минимизации псевдобудевых функций с положительными коэффициентами при нелинейных членах. Более того, установлена эквивалентность этих задач.

Пусть для вектора $g_i, i \in I$ известна перестановка $i_1, \ldots, i_m$, задающая упорядочение элементов

$$g_{i_1} \leq g_{i_2} \leq \cdots \leq g_{i_m}.$$

Положим

$$\Delta g_0 = g_{i_1};$$

$$\Delta g_l = g_{i_{l+1}} - g_{i_l}, 1 \leq l < m;$$

$$\Delta g_m = g_{i_m}.$$

Для любого вектора $z_i \in \{0, 1\}, i \in I, z \neq (1, \ldots, 1)$, справедливы следующие равенства [9, 10]:

$$\min_{i | z_i = 0} g_i = \Delta g_0 + \sum_{l=1}^{m-1} \Delta g_l z_{i_1} \ldots z_{i_l};$$

$$\max_{i | z_i = 0} g_i = \Delta g_m - \sum_{l=1}^{m-1} \Delta g_{m-l} z_{i_{m-l+1}} \ldots z_{i_m}.$$

Для j-го столбца матрицы (c_{ij}) введем перестановку $i_1^j, \ldots, i_m^j$ такую, что

$$c_{i_1^j j} \le c_{i_2^j j} \le \cdots \le c_{i_m^j j}.$$

Представим целевую функцию задачи размещения в виде псевдобулевой функции:

$$b(z) = \sum_{i \in I} f_i(1 - z_i) + \sum_{j \in J} \sum_{l=0}^{m-1} \Delta c_{lj} z_{i_1^j} \ldots z_{i_l^j}.$$

Известно [9, 10], что задача минимизации псевдобулевой функции $b(z), z \ne (1, \ldots, 1)$, и простейшая задача размещения эквивалентны. Для оптимальных решений z^*, S^* этих задач справедливы соотношения $z_i^* = 0 \Leftrightarrow i \in S^*, i \in I$, и значения целевых функций на этих решениях совпадают.

Рассмотрим следующий пример: $I = J = \{1, 2, 3\}$,

$$f_i = \begin{pmatrix} 10 \\ 10 \\ 10 \end{pmatrix}, \qquad c_{ij} = \begin{pmatrix} 0 & 3 & 10 \\ 5 & 0 & 0 \\ 10 & 20 & 7 \end{pmatrix}.$$

Соответствующая псевдобулева функция имеет вид

$$b(z) = 10(1 - z_1) + 10(1 - z_2) + 10(1 - z_3) + (5z_1 + 5z_1z_2) + (3z_2 + 17z_1z_2) +$$

$$(7z_2 + 3z_2z_3) = 15 + 5(1 - z_1) + 0(1 - z_2) + 10(1 - z_3) + 22z_1z_2 + 3z_2z_3.$$

Восстановим по этой псевдобулевой функции задачу размещения. Для новой задачи имеем $I' = I, J' = \{1, 2\}$,

$$f_i' = \begin{pmatrix} 5 \\ 0 \\ 10 \end{pmatrix}, \qquad c_{ij}' = \begin{pmatrix} 0 & 3 \\ 0 & 0 \\ 22 & 0 \end{pmatrix}.$$

Новая задача имеет меньше клиентов, $|J'| < |J|$. Более того, $f_2' = 0$, следовательно, в оптимальном решении второе предприятие можно считать открытым. Другими словами, новая задача имеет меньшую размерность и эквивалентна исходной.

Итак, по простейшей задаче размещения можно построить эквивалентную задачу минимизации псевдобулевой функции с положительными коэффициентами при нелинейных членах. Разные примеры могут давать одну и ту же функцию. Значит, прежде чем решать задачу размещения, можно сначала найти эквивалентную ей задачу, но с меньшей

размерностью. Поиск такой задачи может быть осуществлен за полиномиальное время. Рассмотрим произвольную псевдобулеву функцию $b(z)$ с положительными коэффициентами при нелинейных членах

$$b(z) = \sum_{i \in I} \alpha_i (1 - z_i) + \sum_{l \in L} \beta_l \prod_{i \in I_l} z_i,$$

где $\alpha_i \geq 0, i \in I$ и $\beta_l > 0, I_l \subset I, l \in L$.

Теорема 1 *Для задачи минимизации псевдобулевой функции $b(z)$ с положительными коэффициентами при нелинейных членах эквивалентная ей простейшая задача размещения с минимальным числом клиентов может быть найдена с полиномиальной трудоемкостью от $|L|$ и $|I|$.*

Доказательство. Заметим, что семейство подмножеств $\{I_l\}_{l \in L}$ множества I с отношением порядка $I_{l'} < I_{l''} \Leftrightarrow I_{l'} \subset I_{l''}$ образует частично упорядоченное множество. Любую последовательность подмножеств $I_{l_1} < I_{l_2} < \cdots < I_{l_K}$ называют *цепью*. Произвольное разбиение семейства $\{I_l\}_{l \in L}$ на непересекающиеся цепи порождает матрицу транспортных затрат (c_{ij}) простейшей задачи размещения. Каждый элемент такого разбиения соответствует одному клиенту. Требование найти матрицу с минимальным числом клиентов означает найти разбиение частично упорядоченного множества на минимальное число непересекающихся цепей. Последняя задача может быть решена за полиномиальное время с помощью конструктивного доказательства теоремы Дилворта [194]. □

Задача минимизации функции $b(z)$ эквивалентна простейшей задаче размещения, но обладает новыми свойствами. Например, рассмотрим задачу минимизации этой функции для непрерывных переменных $z_i \in [0, 1], i \in I$. Для простейшей задачи размещения такой переход связан с появлением разрыва целочисленности, который может оказаться сколь угодно близким к единице [163]. Для функции $b(z)$ этот разрыв равен нулю! Более точно, среди оптимальных решений задачи минимизации функции $b(z)$ с непрерывными переменными $z_i \in [0, 1], i \in I$ существует целочисленное оптимальное решение [96].

1.2 Многостадийная задача размещения

Рассмотрим теперь более сложную ситуацию, когда для производства продукции требуется не одно, а несколько предприятий. Будем предпо-

лагать, что имеется k типов предприятий и для производства продукции требуются предприятия каждого типа. Прежде чем попасть к потребителю продукция проходит много стадий обработки. Пусть, как и прежде, множество I задает возможные пункты размещения производства. В каждом пункте можно разместить только одно предприятие заданного типа. Множество I разбивается на k непересекающихся подмножеств $I = I_1 \cup \cdots \cup I_k$. Подмножество I_l задает пункты размещения предприятий l-го типа. Обозначим через P множество технологических цепочек вида $p = \{i_1, \ldots, i_k\}$, согласно которым продукция последовательно проходит обработку на предприятиях $i_1, \ldots, i_k$, где $i_l \in I_l, l = 1, \ldots, k$, и пусть $P_i = \{p \in P | i \in p\}, i \in I$. Для каждой пары $p \in P$, $j \in J$ считаем известной величину расходов $c_{pj} \geq 0$, связанных с производством продукции по технологической цепочке p и доставкой ее j-му клиенту.

Введем переменные

$$x_{pj} = \begin{cases} 1, & \text{если клиент } j \text{ снабжается по технологической цепочке } p, \\ 0 & \text{в противном случае.} \end{cases}$$

Многостадийная задача размещения (МЗР) состоит в отыскании такого варианта размещения предприятий и выбора на их основе подмножества технологических цепочек, чтобы с минимальными суммарными затратами удовлетворить запросы всех потребителей. С использованием введенных обозначений задача может быть записана следующим образом [10, 49, 106, 203]: найти

$$\min \Big\{ \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{p \in P} c_{pj} x_{pj} \Big\}$$

при условиях

$$\sum_{p \in P} x_{pj} = 1, \quad j \in J,$$

$$\sum_{p \in P_i} x_{pj} \leq x_i, \quad i \in I,\ j \in J,$$

$$x_{pj},\ x_i \in \{0, 1\}, \quad i \in I,\ p \in P,\ j \in J.$$

Целевая функция задачи определяет суммарные затраты на открытие предприятий, производство продукции и ее доставку потребителям. Первое ограничение гарантирует удовлетворение спроса всех потребителей. Второе ограничение позволяет обслуживать клиентов только из открытых предприятий.

Сформулированная задача является NP-трудной в сильном смысле и тесно связана с задачами стандартизации [10], задачей минимизации полиномов от булевых переменных [11] и двухуровневой задачей выбора номенклатуры изделий [24]. Если каждая цепочка $p \in P$ содержит только одно предприятие, то получаем простейшую задачу размещения. В работах [9, 10, 203] вместо условия $\sum_{p \in P_i} x_{pj} \leq x_i, \quad i \in I, \ j \in J,$ используется условие

$$x_{pj} \leq x_i, \quad p \in P_i, i \in I, \ j \in J,$$

что приводит к более слабой линейной релаксации. Такой же эффект получается введением переменных

$$y_p = \begin{cases} 1, & \text{если выпускается продукция по технологической цепочке } p, \\ 0 & \text{в противном случае,} \end{cases}$$

и заменой условия $\sum_{p \in P_i} x_{pj} \leq x_i, \quad i \in I, \ j \in J,$ на ограничения

$$y_p \geq x_{pj}, \quad j \in J, p \in P,$$

$$x_i \geq y_p, \quad p \in P_i, i \in I.$$

Новая формулировка дает ту же оценку линейного программирования, что и предыдущая, но эта оценка может оказаться в произвольное число раз хуже исходной [39].

Заметим, что в математической постановке задачи нигде не отражается тот факт, что множество I разбито на непересекающиеся подмножества. Множество P явным образом задает все технологические цепочки и, вообще говоря, может иметь произвольную структуру. Например, оно может содержать цепочки разной длины и с несколькими вхождениями одного и того же предприятия. В работе [22] с величинами $c_{pj}, \ p \in P, j \in J,$ связывались затраты на транспортировку продукции между предприятиями и затраты на доставку ее потребителю. Сюда же включались и затраты на обработку и выпуск продукции каждым предприятием из данной технологической цепочки. Легко видеть, что такой подход позволяет моделировать ситуацию произвольных цепочек с любым маршрутом продукции по предприятиям. В этом смысле ситуация близка к задачам Job Shop из теории расписаний, где допускается произвольная последовательность обработки деталей на станках.

Предложенная модель не всегда удобна при решении практических задач, так как множество P может оказаться экспоненциально большим.

В работе [10] рассматривается возможность неявного задания этого множества. Такой подход существенно экономит память, необходимую для хранения исходной информации, но лишает модель гибкости при учете производственно–транспортных затрат и свободы в формировании цепочек. В дальнейшем будет рассматриваться только модель с явным заданием множества P, которое будем считать частью исходных данных.

Представим МЗР в виде задачи безусловной оптимизации:

$$\min_{y_p} \Big\{ \sum_{i\in I} f_i \max_{p\in P_i}\{y_p\} + \sum_{j\in J} \min_{p\in P}\{c_{pj} \mid y_p = 1\} \Big\}.$$

Используя представление минимума и максимума через произведение переменных, получаем псевдобулеву функцию

$$B(z) = \sum_{i\in I} f_i - \sum_{i\in I} f_i \prod_{p\in P_i} z_p + \sum_{j\in J} \sum_{l=0}^{m-1} \Delta c_{lj} z_{i_1^j} \dots z_{i_l^j}.$$

Заметим, что функция $B(z)$ имеет как положительные, так и отрицательные коэффициенты при нелинейных членах. Известно [9, 10], что многостадийная задача размещения эквивалентна задаче минимизации псевдобулевой функции $B(z)$, $z \neq (1,\dots,1)$. Для оптимальных решений z^*, y^* этих задач справедливы соотношения $z_p^* = 1 - y_p^*$, $p \in P$ и значения целевых функций на этих решениях совпадают. Таким образом, многостадийная задача размещения сводится к задаче минимизации псевдобулевой функции, а по произвольной псевдобулевой функции, по аналогии с Теоремой 1, можно построить пример многостадийной задачи размещения с минимальным числом клиентов и исходным множеством открываемых предприятий.

Теорема 2 *Для любого примера многостадийной задачи размещения можно за полиномиальное время от размерности задачи построить эквивалентный пример с минимальным числом клиентов.*

Доказательство. Представим задачу МЗР в виде задачи минимизации функции $B(z)$. Каждый элемент множества I будет порождать в общем случае нелинейное отрицательное слагаемое (при $|P_i| > 1$). Производственно транспортные затраты (c_{pj}) будут порождать неотрицательные слагаемые. Восстановим пример задачи МЗР по полученной псевдобулевой функции. Отрицательные слагаемые снова будут порождать элементы множества I. Оно сохранит свою мощность. При восстановлении матрицы производственно-транспортных затрат снова воспользуемся конструктивным доказательством теоремы Дилворта и найдем минимальное

число непересекающихся цепей, на которое разбивается соответствующее частично-упорядоченное множество. Каждая такая цепь порождает элемент множества J. □

1.3 Задачи размещения с предпочтениями клиентов

До сих пор предполагалось, что имеется одно лицо, принимающее решение (ЛПР). Это руководитель фирмы, который стремится минимизировать суммарные затраты на организацию производства и обслуживание клиентов. Однако в рыночных условиях клиент имеет возможность сам выбирать поставщиков продукции, исходя из собственных предпочтений [23, 142]. Он не обязан минимизировать производственно–транспортные затраты фирмы.

Пусть матрица (g_{ij}) задает предпочтения клиентов на множестве I. Если $g_{i_1 j} < g_{i_2 j}$, $i_1 \neq i_2$, то j-й клиент предпочитает предприятие i_1. Для упрощения модели будем предполагать, что в каждом столбце матрицы g_{ij} все элементы различные. В противном случае придется рассматривать оптимистические и пессимистические стратегии и вводить понятия кооперативной и некооперативной игры [14]. Итак, цель ЛПР по прежнему состоит в выборе подмножества $S \subseteq I$, которое позволяет обслужить всех клиентов с минимальными суммарными затратами, но теперь ЛПР вынужден учитывать предпочтения клиентов по выбору поставщиков. Математическая модель для такого случая может быть представлена в виде задачи двухуровневого программирования [2, 23]: найти

$$\min_{x_i} \Big\{ \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x^*_{ij}(x_i) \Big\}$$

при условиях

$$x_i \in \{0, 1\}, \quad i \in I,$$

где $x^*_{ij}(x_i)$ — оптимальное решение задачи клиентов: найти

$$\min_{x_{ij}} \sum_{j \in J} \sum_{i \in I} g_{ij} x_{ij}$$

при условиях

$$\sum_{i \in I} x_{ij} = 1, \qquad j \in J,$$

$$x_{ij} \leq x_i, \qquad i \in I, j \in J,$$

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

В этой задаче целевая функция по-прежнему задает суммарные затраты на открытие предприятий и обслуживание клиентов. Отличие состоит в том, что теперь множество допустимых решений задается как требованием целочисленности переменных $x_i, i \in I$ (что соответствует условию $S \subseteq I$), так и вспомогательной оптимизационной задачей — задачей клиентов по выбору поставщиков. При решении вспомогательной задачи вектор x_i считается известным.

Существуют несколько способов сведения данной двухуровневой задачи к задаче целочисленного линейного программирования [14, 144]. Заметим, что для каждого $j \in J$ важен только порядок элементов g_{ij}, а не их числовые значения. Упорядочим элементы j-го столбца по возрастанию

$$g_{i_1 j} < g_{i_2 j} < ... < g_{i_m j}$$

и положим $S_{ij} = \{l \in I |\ g_{lj} < g_{ij}\},\ i \in I$. Оптимальное решение $x^*_{ij}(x_i)$ внутренней задачи клиентов обладает следующим свойством:

$$x^*_{ij} = 1 \Rightarrow x_l = 0, \quad l \in S_{ij}.$$

Используя это соотношение, исходную двухуровневую задачу можно представить в следующем виде: найти

$$\min \sum_{i \in I} f_i x_i + \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij}$$

при условиях:

$$x_{ij} + x_l \leq 1, \quad l \in S_{ij}, i \in I, j \in J,$$

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

$$x_{ij} \leq x_i, \qquad i \in I, j \in J,$$

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

Первое неравенство в этой задаче гарантирует, что значения x_{ij} будут соответствовать оптимальному решению задачи клиентов. Остальные ограничения совпадают с ограничениями простейшей задачи размещения. Число переменных задачи равно $m + nm$, то есть их столько же, сколько в простейшей задаче размещения, но число ограничений на $O(m^2 n)$ больше, что не позволяет решать эту задачу стандартным программным обеспечением даже при средней размерности задачи. Следуя

[23, 144], дополнительные ограничения можно записать в другой эквивалентной форме:

$$\sum_{l \in S_{ij}} x_l \le |S_{ij}|(1 - x_{ij}), \quad i \in I, j \in J,$$

или

$$x_i \le x_{ij} + \sum_{l \in S_{ij}} x_l, \quad i \in I, j \in J,$$

или

$$x_i \le x_{ij} + \sum_{l \in S_{ij}} x_{lj}, \quad i \in I, j \in J.$$

Можно показать, что последнее неравенство приводит к лучшей линейной релаксации, чем три других [2].

Рассмотрим частный случай задачи, когда стоимости открытия предприятий равны нулю, $f_i = 0,\ i \in I$. Для простейшей задачи размещения это тривиальный случай, оптимальное решение находится за линейное время. Если же клиенты имеют собственные предпочтения, то задача остается NP-трудной, а разрыв целочисленности может оказаться сколь угодно близким к единице [23, 144].

Известно, что задача размещения с предпочтениями клиентов может быть сведена к задаче минимизации псевдобулевой функции. Пусть перестановка $i_1, i_2, \ldots, i_m$ задает упорядочение элементов j-го столбца матрицы (g_{ij}). Для $j \in J$ положим

$$\nabla c_{i_1 j} = c_{i_1 j},$$

$$\nabla c_{i_l j} = c_{i_l j} - c_{i_{l-1} j}, \quad 1 < l \le m,$$

и определим псевдобулеву функцию

$$B(z) = \sum_{i \in I} f_i(1 - z_i) + \sum_{j \in J} \sum_{i \in I} \nabla c_{ij} \prod_{l \in S_{ij}} z_l.$$

Задача размещения с предпочтениями клиентов эквивалентна задаче минимизации псевдобулевой функции $B(z),\ z \ne (1, \ldots, 1)$. Оптимальные решения z^*, x^* этих задач связаны соотношением $z_i^* = 1 - x_i^*,\ i \in I$, и значения целевых функций на этих решениях совпадают [9, 10].

Заметим, что коэффициенты ∇c_{ij} могут быть как положительными, так и отрицательными. Другими словами, для задачи минимизации

псевдобулевой функции можно построить пример задачи размещения с предпочтениями клиентов и наоборот. Более того, как указано в предыдущем разделе, для задачи минимизации псевдобулевой функции можно построить пример многостадийной задачи размещения. Следовательно, пользуясь псевдобулевыми функциями, можно преобразовать любой пример задачи размещения с предпочтениями клиентов в пример многостадийной задачи и наоборот. Используя тот же подход, что и в Теореме 1, для задачи минимизации псевдобулевой функции можно за полиномиальное время получить пример задачи размещения с предпочтениями клиентов с минимальным числом клиентов.

Утверждение 1. Для любого примера задачи размещения предприятий с предпочтениями клиентов можно за полиномиальное время от размерности задачи построить эквивалентный пример с минимальным числом клиентов.

Доказательство. Построим псевдобулеву функцию $B(z)$, приведем подобные члены. При определении частично-упорядоченного множества не будем обращать внимание на знак величины ∇c_{ij}. Семейство подмножеств $\{S_{ij}\}_{i\in I, j\in J}$ с отношением порядка $S_{ij} < S_{i'j'} \Leftrightarrow S_{ij} \subset S_{i'j'}$ порождает частично упорядоченное множество. Его разбиение на минимальное число непересекающихся цепей дает пример задачи с минимальным числом клиентов. □

В заключение этого раздела укажем еще на одну модель, которая является обобщением двух последних моделей, а именно, на многостадийную модель с предпочтениями клиентов [22]. Используя уже введенные обозначения, получаем следующую задачу двухуровневого программирования (ДМЗР): найти

$$\min\Big\{\sum_{i\in I} f_i x_i + \sum_{j\in J}\sum_{p\in P} c_{pj} x^*_{pj}(x_i)\Big\}$$

при условии:

$$x_i \in \{0,1\}, \quad i \in I,$$

где $x^*_{pj}(x_i)$ — оптимальное решение задачи: найти

$$\min \sum_{j\in J}\sum_{p\in P} g_{pj} x_{pj}$$

при условиях:

$$\sum_{p\in P} x_{pj} = 1, \quad j \in J,$$

$$\sum_{p\in P_i} x_{pj} \ \leq x_i, \quad i \in I,\ j \in J,$$

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

Теорема 3 *Задачи ДМЗР и МЗР полиномиально сводятся друг к другу.*

Доказательство. Сведение задачи МЗР к ДМЗР очевидно. Покажем обратное сведение. Следуя [24], для $j \in J$ введем перестановку $(p_1, p_2, \ldots, p_{m_0})$, $m_0 = |P|$ такую, что $g_{p_1 j} < g_{p_2 j} < \cdots < g_{p_{m_0} j}$, и положим

$$a_{p_k j} = c_{p_1 j} + \sum_{l=1}^{k-1} \min\{0; c_{p_{l+1} j} - c_{p_l j}\},\ k = 1, \ldots, m_0,$$

$$b_{p_k j} = \sum_{l=1}^{k-1} \max\{0; c_{p_{l+1} j} - c_{p_l j}\},\ k = 1, \ldots, m_0.$$

Тогда для непустого множества $S \subseteq P$ и вектора $y^*_{pj} \in \{0;1\}$, определенного равенством

$$y^*_{pj} = \begin{cases} 1, & \text{если } g_{pj} = \min_{p\in S} g_{pj}, \\ 0 & \text{в противном случае,} \end{cases} \quad j \in J,$$

получаем

$$\max_{p\in S} a_{pj} + \min_{p\in S} b_{pj} = \sum_{p\in P} c_{pj} y^*_{pj}.$$

Полагая $f_{pi} = f_i$, если $p \in P_i$ и $f_{pi} = 0$ в противном случае, получаем сведение задачи ДМЗР к задаче выбора подмножества строк пары матриц:

$$\min_{S\subseteq P, S\neq\emptyset} \left\{ \sum_{i\in I} \max_{p\in S} f_{pi} + \sum_{j\in J} \max_{p\in S} a_{pj} + \sum_{j\in J} \min_{p\in S} b_{pj} \right\}.$$

Эквивалентность же задачи выбора подмножества строк пары матриц и задачи МЗР показана в [9, 10]. □

Таким образом, задача ДМЗР также может быть представлена как задача минимизации псевдобулевой функции и, следовательно, переформулирована в виде многостадийной задачи без предпочтений клиентов или задачи с предпочтениями клиентов, но не многостадийной, а одностадийной.

Добавим в систему ограничений верхнего уровня условие

$$\sum_{i \in I} x_i = p.$$

Полученную задачу назовем двухуровневой задачей о p-медиане (ДМ). Повторяя доказательство теоремы 3, получаем, что задача ДМ также может быть представлена в виде задачи минимизации псевдобулевой функции с соответствующим дополнительным ограничением:

$$\sum_{i \in I} z_i = m - p,$$

где $z_i = 1 - x_i, i \in I$. Отметим, что коэффициенты этой псевдобулевой функции при нелинейных членах могут иметь произвольный знак.

1.4 Антагонистические размещения

Рассмотрим теперь ситуацию, когда две фирмы последовательно принимают решения о размещении предприятий. Сначала на рынок выходит первая фирма (Лидер) и открывает свое подмножество предприятий $S_0 \subseteq I$. Затем, зная это решение, конкурирующая фирма (Конкурент) открывает собственные предприятия, подмножество $S_1 \subset I, S_0 \cap S_1 = \emptyset$. Каждый клиент выбирает из множества открытых предприятий $S_0 \cup S_1$ одно предприятие согласно собственным предпочтениям. Если обслуживание j-го клиента приносит доход $w_j > 0$, Лидер открывает p предприятий, а Конкурент — r предприятий, то в зависимости от размещения этих предприятий рынок (множество клиентов) будет как-то разделен между двумя фирмами. Каждая фирма стремится максимизировать свою долю рынка. Получаем игру двух лиц с противоположными интересами. Игроки неравноправны. Сначала делает ход первый игрок. Он может размещать предприятия в любом месте, и это может давать ему преимущество. Затем делает ход второй игрок, уже зная ход первого игрока. Получаем игру Штакельберга, в которой требуется максимизировать долю рынка (суммарный доход) первого игрока. Введем переменные задачи:

Лидер:

$$x_i = \begin{cases} 1, & \text{если Лидер в пункте } i \text{ открывает предприятие,} \\ 0 & \text{в противном случае,} \end{cases}$$

Конкурент:

$$y_i = \begin{cases} 1, & \text{если Конкурент в пункте } i \text{ открывает предприятие,} \\ 0 & \text{в противном случае,} \end{cases}$$

Клиенты:

$$u_j = \begin{cases} 1, & \text{если клиент } j \text{ обслуживается из предприятия Лидера,} \\ 0, & \text{если клиент } j \text{ обслуживается из предприятия Конкурента.} \end{cases}$$

При заданном векторе $x_i \in \{0,1\}, i \in I$, определим множество

$$I_j(x) = \{i \in I \,|\, g_{ij} < \min_{l \in I}(g_{lj} \,|\, x_l = 1)\}, \quad j \in J.$$

Это множество задает пункты размещения предприятий, позволяющие Конкуренту *захватить* j-го клиента. С использованием введенных переменных соответствующая задача двухуровневого программирования записывается следующим образом: найти

$$\max_x \sum_{j \in J} w_j u_j^*(x, y^*)$$

при условиях:

$$\sum_{i \in I} x_i = p,$$

$$x_i \in \{0,1\}, \quad i \in I,$$

где $u^*(x, y^*), y^*$ — оптимальное решение задачи Конкурента: найти

$$\max_{y,u} \sum_{j \in J} w_j (1 - u_j)$$

при условиях:

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

$$u_j \le 1 - y_i, \quad i \in I_j(x), j \in J,$$

$$y_i + x_i \le 1, \quad i \in I,$$

$$y_i, u_j \in \{0,1\}, \quad i \in I, j \in J.$$

Целевая функция задачи задает суммарный доход Лидера. Множество допустимых решений описывается с помощью вспомогательной задачи

Конкурента. При решении задачи Конкурента вектор $x_i,\ i \in I$, и множества $I_j(x),\ j \in J$, считаются известными. На сегодняшний день эффективные методы решения данной задачи неизвестны. Первые шаги в этом направлении сделаны в [190], где предлагается схема неявного перебора. В [182] исследуется частный случай задачи, когда Конкурент размещает только одно предприятие. Заметим, что данная модель легко обобщается на случай, когда на рынке уже размещены какие-то предприятия и Лидер вынужден *бороться* за клиентов не только с Конкурентом, но и с этими уже открытыми предприятиями.

В задаче Конкурента предполагалось, что нельзя открывать предприятия в тех местах, где уже есть предприятия Лидера, т.е. $S_0 \cap S_1 = \emptyset$. От этого условия можно отказаться. Удаление ограничения $y_i + x_i \leq 1,\ i \in I$ из задачи Конкурента не меняет его оптимального решения. Конкуренту невыгодно открывать такие предприятия, так как это не дает дополнительных клиентов. Однако ситуация может измениться, если переопределить множества $I_j(x)$ и положить

$$I_j(x) = \{i \in I | g_{ij} \leq \min_{l \in I}(g_{lj} | x_l = 1)\}, \quad j \in J.$$

В этом случае Лидер будет терять клиента даже в том случае, когда Конкурент открывает новое предприятие на том же расстоянии от клиента, что и Лидер. Грубо говоря, получаем модель с "любознательными" клиентами, которые при прочих равных условиях тянутся к новому предприятию. В [107] рассмотрено три возможных случая: любознательные клиенты, консервативные клиенты и промежуточный вариант, когда клиенты в половине случаев ведут себя как любознательные, а в остальных случаях — как консервативные. В [139] отмечено, что случай с любознательными клиентами не представляет интерес, т.к. у Конкурента появляется возможность открыть свои предприятия в том же месте, что и Лидер, и отобрать у него весь рынок. В дальнейшем будет рассматриваться только задача с консервативными клиентами.

Покажем связь предложенной задачи двухуровневого программирования с минимаксными задачами для псевдобулевых функций. Заметим, что $u_j^*(x) = \prod_{i \in I_j(x)}(1 - y_i^*)$. Тогда задачу можно представить следующим образом: найти

$$\max \sum_{j \in J} w_j \prod_{i \in I_j(x)} (1 - y_i^*(x))$$

при условиях:

$$\sum_{i\in I} x_i = p, \quad x_i \in \{0,1\},$$

где $y^*(x)$ — оптимальное решение задачи Конкурента: найти

$$\min \sum_{j\in J} w_j \prod_{i\in I_j(x)} (1-y_i)$$

при условиях:

$$\sum_{i\in I} y_i = r, \quad y_i \in \{0,1\}.$$

Таким образом, получаем задачу двухуровневого программирования для псевдобулевой функции

$$P(x,y) = \sum_{j\in J} w_j \prod_{i\in I_j(x)} (1-y_i),$$

или максиминную задачу вида:

$$\max_x \min_y \{P(x,y) \mid \sum_{i\in I} x_i = p, \quad \sum_{i\in I} y_i = r, \quad x_i, y_i \in \{0,1\}\}.$$

Если не менять целевую функцию Конкурента и ввести новую псевдобулеву функцию

$$P'(x,y) = \sum_{j\in J} w_j - P(x,y),$$

то исходную задачу двухуровневого программирования можно представить как минимаксную задачу вида:

$$\min_x \max_y \Big\{P'(x,y) \mid \sum_{i\in I} x_i = p, \quad \sum_{i\in I} y_i = r, \quad x_i, y_i \in \{0,1\}\Big\}.$$

В таком виде задача известна в литературе как задача об (r,p)-центроиде [139]. В [138] показано, что поиск допустимого решения такой задачи, т.е. нахождение оптимального решения Конкурента при заданном решении Лидера, является NP–трудной задачей. Там же установлено, что задача о центроиде является NP–трудной даже в том случае, если Конкурент открывает только одно предприятие. В общем случае задача является Σ_2^P-трудной [177]. Полиномиально разрешимые случаи можно найти в [177, 199]. Обзор результатов содержится в [36].

Заметим, что формулировка задачи и содержательный смысл переменных $u_j \in \{0,1\}$ предполагает, что все клиенты будут обслужены либо Лидером, либо Конкурентом. В [182] рассматривается более общая

ситуация, когда у каждого клиента (предприятия) имеется свое множество возможных поставщиков (клиентов). Игроки по-прежнему стремятся максимизировать свою долю рынка, но теперь уже нет гарантии, что все клиенты будут обслужены. В оптимальном решении часть наиболее "удаленных" клиентов могут не попасть в зону обслуживания ни одного из открытых предприятий. Задача уже не сводится к максиминной или минимаксной задаче для указанных псевдобулевых функций, а само понятие оптимального решения требует дополнительных уточнений.

1.5 Задачи стандартизации и унификации

Математическим моделям стандартизации и унификации посвящены десятки статей. В них идет речь о выборе оптимального состава системы технических средств, предназначенных для выполнения заданного круга работ. Толчком для исследований в этой области послужила монография [10], где представлен широкий спектр дискретных оптимизационных задач и методов их решения. На первый взгляд задачи стандартизации и унификации по своему смыслу далеки от задач размещения. Тем не менее, для простейшей задачи выбора ряда изделий одноразового использования после приведения подобных членов и исключения части переменных получаем в точности простейшую задачу размещения. Исследование задачи выбора ряда изделий и комплектующих узлов приводит к многостадийной задаче размещения [10, 21]. Эти две области исследования операций, размещение и унификация, оказываются тесно связанными, хотя и имеют собственную логику развития. В данном разделе обсуждаются четыре модели выбора оптимального состава систем технических средств:

— модель с многоэтапным процессом выполнения работ,
— модель двухуровневой системы технических средств,
— динамическая модель с ограничениями на объемы производства,
— модель с частичным внешним финансированием.

В разные годы эти модели были предметом исследований в рамках прикладных НИР, проводимых в Институте математики СО РАН по заказу различных ведомств, в том числе и Госстандарта.

1.5.1 Задача выбора состава системы технических средств

Пусть множество $I = \{1, \ldots, m\}$ задает исходный ряд образцов технических средств. Он включает в себя номера образцов, которые уже выпускаются серийно, либо могут быть выпущены в ближайшем будущем. Системой технических средств называют произвольный набор образцов из исходного ряда.

Технические средства предназначены для выполнения определенных работ. Совокупность этих работ называют областью применения системы и обозначать множеством $J = \{1, \ldots, n\}$. Считаем, что для каждой работы существуют технические средства, предназначенные для ее выполнения и каждое техническое средство может использоваться только для выполнения работ области применения. Процесс выполнения работ разбит на этапы. На первом этапе выполняются работы из множества $J_1 \subset J$, на втором — из $J_2 \subset J$ и т.д. Будем считать, что

$$J = \bigcup_{l=1}^{L} J_l, \quad J_l \cap J_k = \emptyset, \quad k \neq l, \quad |J_l| = n_l.$$

Задача состоит в том, чтобы найти состав системы, который позволяет выполнить все работы с минимальными суммарными затратами.

Математическая постановка может быть записана в виде задачи линейного частично–целочисленного программирования. Для каждого образца $i \in I$ введем следующие обозначения:

c_i^0 — начальные затраты на научные исследования, опытно–конструкторские работы и подготовку производства;

c_i — затраты на производство одного изделия;

V_i — максимально возможный объем производства изделий;

s_{il} — доля потерь на l-м этапе выполнения работ;

v_i^0 — объем изделий, уже имеющихся в наличии (начальный состав системы);

p_{ij} — количество изделий, требующихся для выполнения j-ой работы;

c_{ij} — затраты на эксплуатацию при выполнении j-ой работы.

p — верхняя граница на число образцов в составе системы;

Переменные задачи имеют следующий смысл:

$$x_i = \begin{cases} 1, & \text{если } i\text{-й образец включен в состав системы,} \\ 0 & \text{в противном случае;} \end{cases}$$

$v_i \geq 0$ — объем производства изделий i-го образца;

$x_{ij} \geq 0$ — доля работы j, выполняемая изделиями i-го образца.

Сформулированная задача выбора состава системы технических средств (ВСС) записывается следующим образом: найти

$$\min \sum_{i \in I} (c_i^0 x_i + c_i v_i + \sum_{j \in J} c_{ij} x_{ij})$$

при условиях:

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

$$\sum_{i \in I} x_i \leq p,$$

$$\sum_{j \in J_l} p_{ij} x_{ij} \leq v_i^0 + v_i - \sum_{\tau=1}^{l-1} s_{i\tau} \sum_{j \in J_\tau} p_{ij} x_{ij}, \quad i \in I, l = 1, \ldots, L,$$

$$0 \leq v_i \leq V_i, \quad i \in I,$$

$$0 \leq x_{ij} \leq x_i, \quad i \in I, j \in J,$$

$$x_i \in \{0, 1\}, \quad i \in I.$$

Целевая функция выражает суммарные затраты на создание и функционирование системы. Первое ограничение гарантирует выполнение всех работ области применения. Второе неравенство ограничивает сверху номенклатуру образцов. Третье неравенство отражает баланс между требующимися изделиями на каждом этапе выполнения работ и имеющимися в наличии изделиями. В левой части этого неравенства стоит выражение для числа изделий i-го образца, требующегося на l-м этапе, а в правой части — выражение для имеющегося в наличии их числа с учетом потерь на предшествующих этапах. Следующее ограничение устанавливает верхнюю границу на возможные объемы производства изделий. Последнее неравенство запрещает использование образцов, не включенных в состав системы.

Заметим, что при одном этапе выполнения работ, $L = 1$, нулевом начальном составе системы и неограниченных объемах производства получаем простейшую задачу размещения с дополнительным ограничением на число открываемых предприятий. Если начальные затраты равны нулю, $c_i^0 = 0,\ i \in I$, то приходим к задаче о p-медиане. Наличие начального состава системы позволяет выделить в качестве подзадачи известную

задачу размещения с ограничением на объемы продукции, выпускаемой каждым предприятием [111]. Таким образом, представленная задача является обобщением нескольких известных задач размещения и, как следствие, является NP-трудной в сильном смысле.

1.5.2 Двухуровневые системы технических средств

Предположим теперь, что для выполнения работ технические средства объединяются в комплекты, которые могут рассматриваться как новые сложные технические средства, имеющие в своем составе унифицированные составные части. Процесс выполнения работ по-прежнему разбит на этапы. На каждом этапе определенная доля комплектов выходит из строя и в дальнейшем процессе выполнения работ не участвует. Задача состоит в том, чтобы найти состав системы комплектов технических средств, который позволил бы выполнить все работы с минимальными суммарными затратами.

Обозначим через $K = \{1, 2, \ldots, q\}$ множество различных по составу комплектов. Для $k \in K$ множество $I_k \subseteq I$ будет задавать состав комплекта k-го типа, а для $i \in I$ множество $K_i \subseteq K$ — перечень комплектов, имеющих в своем составе технические средства i-го образца. Для каждого комплекта k-го типа, $k \in K$, считаем известными следующие параметры:

v_k° — число комплектов, уже имеющихся в составе системы;

c_k° — стоимость разработки комплекта;

d_i — стоимость разработки технического средства i-го образца;

c_k — стоимость производства одного комплекта, включая стоимость входящих в его состав технических средств;

V_k — верхняя граница объемов производства комплектов;

p_{kj} — число комплектов, требующихся для выполнения j-й работы;

c_{kj} — стоимость выполнения комплектами j-й работы;

s_{kl} — доля комплектов, выходящих из строя на l-м этапе.

Введем следующие переменные:

$$z_k = \begin{cases} 1, & \text{если комплекты } k\text{-го типа включены в состав системы,} \\ 0 & \text{в противном случае;} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{если средства } i\text{-го образца включены в состав системы,} \\ 0 & \text{в противном случае;} \end{cases}$$

$v_k \geq 0$ — объем производства комплектов k-го типа;

$x_{kj} \geq 0$ — доля j-й работы, выполняемая комплектами k-го типа.

С использованием введенных обозначений задачу выбора состава двухуровневой системы технических средств (В2СС) можно записать в виде задачи частично-целочисленного программирования: найти

$$\min_{x,y,z,v} \Big(\sum_{k \in K} \Big(c_k^{\circ}\ z_k + c_k\ v_k + \sum_{j \in J} c_{kj}\ x_{kj} \Big) + \sum_{i \in I} d_i\ y_i \Big)$$

при условиях:

$$\sum_{k \in K} x_{kj} = 1, \quad j \in J,$$

$$\sum_{j \in J_l} p_{kj} x_{kj} \leq v_k^{\circ} z_k + v_k - \sum_{l'=1}^{l-1} s_{kl'} \sum_{j \in J_{l'}} p_{kj} x_{kj}, \quad k \in K, \quad l = 1, \ldots, L,$$

$$0 \leq v_k \leq V_k z_k, \quad k \in K,$$

$$0 \leq x_{kj} \leq z_k, \quad k \in K, \quad j \in J,$$

$$y_i \geq z_k, \quad k \in K_i, \quad i \in I,$$

$$y_i, z_k \in \{0, 1\}, \quad i \in I, \quad k \in K.$$

Целевая функция отражает суммарные затраты на разработку, производство и эксплуатацию комплектов технических средств. Первое ограничение гарантирует выполнение всех работ. Второе ограничение выражает связь между объемами производства комплектов и объемами работ. Для каждой пары (k, l) левая часть неравенства задает требующееся количество комплектов k-го типа на l-м этапе выполнения работ. Правая часть задает количество комплектов k-го типа, имеющихся в наличии к началу l-го этапа с учетом потерь на предыдущих этапах. Третье ограничение устанавливает верхнюю границу на возможные объемы производства комплектов. Последние два ограничения допускают использование комплектов и технических средств только при включении их в состав системы.

Первые математические модели выбора состава двухуровневых систем технических средств были рассмотрены в [7]. Наиболее полно такие системы изучались в монографии [10], где, в частности, установлено сведение наиболее простой линейной задачи выбора состава двухуровневых систем к задаче минимизации полинома от булевых переменных. Большинство статей (см., например, [11, 19, 21, 88, 203]) посвящено исследованию частного случая рассматриваемой задачи — многостадийной

задаче размещения, которая получается из задачи B2CC при одном этапе выполнения работ.

1.5.3 Динамическая задача с ограничениями на мощности

Предположим, что в течение определенного планового периода необходимо выполнить заданную совокупность работ. Для выполнения работ имеется система технических средств. Состав системы может меняться за счет пополнения новыми образцами и изъятия части изделий по причинам их морального или физического старения. Динамическая задача выбора оптимального состава системы состоит в том, чтобы найти такой вариант развития системы, который доставлял бы минимум суммарных затрат на разработку, производство и эксплуатацию технических средств при условии выполнения каждый год заданного списка работ.

Рассмотрим плановый период, в течение которого система технических средств должна выполнять определенные объемы работ. Пусть l — продолжительность планового периода, для определенности измеренная в годах, $T = \{1, \ldots, l\}$. Множество рассматриваемых образцов технических средств по прежнему будем обозначать через $I = \{1, \ldots, m\}$, а множество работ, подлежащих выполнению в течение всего интервала планирования, — через $J = \{1, \ldots, n\}$. Будем предполагать, что множество J разбито на непересекающиеся подмножества $J_t, t \in T$. Для каждого $j \in J$ известен номер $t(j)$ такой, что $j \in J_{t(j)}$. Для изделий i-го образца считаем заданными следующие величины:

c^0_{it} — стоимость разработки изделия к t-му году,
c_{it} — стоимость производства одного изделия в t-м году,
c_{ij} — стоимость выполнения j-й работы,
p_{ij} — число изделий, требующихся для выполнения j-й работы,
ρ_i — длительность эксплуатации одного изделия,
v^0_{it} — начальный состав системы к t-му году,
V_{it} — верхняя граница объемов производства изделий в t-м году.

Переменные задачи имеют следующий смысл:

$$z_{it} = \begin{cases} 1, & \text{если в } t\text{-м году закончена разработка изделий } i\text{-го образца,} \\ 0 & \text{в противном случае,} \end{cases}$$

$v_{it} \geq 0$ — объем производства изделий i-го образца в t-м году,

$x_{ij} \geq 0$ — доля j-й работы, выполняемая изделиями i-го образца.

С помощью введенных обозначений, динамическая задача выбора оптимального состава системы (ДЗ) записывается следующим образом: найти

$$\min \sum_{t \in T} \sum_{i \in I} \left\{ c^0_{it} z_{it} + c_{it} v_{it} + \sum_{j \in J_t} c_{ij} x_{ij} \right\}$$

при условиях:

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

$$\sum_{j \in J_t} p_{ij} x_{ij} \leq v^0_{it} + \sum_{\tau = t - \rho_i + 1}^{t} v_{i\tau}, \quad i \in I, t \in T,$$

$$0 \leq v_{it} \leq V_{it} \max_{\tau \leq t} z_{i\tau}, \quad i \in I, t \in T,$$

$$0 \leq x_{ij} \leq \max_{\tau \leq t(j)} z_{i\tau}, \quad i \in I, j \in J,$$

$$z_{it} \in \{0, 1\}, \quad i \in I, t \in T.$$

Целевая функция задачи выражает суммарные затраты на разработку, производство и эксплуатацию технических средств в течение планового периода. Первое условие гарантирует выполнение всех работ. Второе условие ограничивает возможности выполнения работ имеющимися в наличии изделиями. Третье условие устанавливает верхнюю границу на возможные объемы производства. Четвертое условие требует внесения затрат на разработку новых образцов при включении их в состав системы. Сформулированная задача относится к числу NP-трудных задач дискретной оптимизации, так как является обобщением простейшей задачи размещения.

Первые постановки динамических задач выбора оптимального состава системы изучались в работе [18], где, в частности, найдено точное решение задачи в предположении, что в определенные моменты времени может проводиться полная замена старой системы технических средств на новую систему. Эти результаты вошли в монографию [10], в которой наиболее полно представлена эта проблематика.

Одним из частных случаев сформулированной задачи является динамическая задача о размещении [67], для решения которой предложены как приближенные [151], так и точные алгоритмы [113, 204], основанные на вычислительной схеме метода ветвей и границ. В более простой постановке данная задача рассматривалась в [12], где также использовались

идеи метода ветвей и границ. Однако специфика данной задачи, а именно нетривиальная подзадача линейного программирования, получаемая при фиксированных целочисленных переменных, значительно затрудняет использование предложенного подхода.

1.5.4 Задача выбора состава системы с частичным внешним финансированием

В этом разделе рассматривается двухуровневая математическая модель, основной особенностью которой является покрытие части расходов на организацию производства средствами инвестора. Он выделяет кредит и может получить любые изделия из числа произведенных по заранее согласованным ценам. На верхнем уровне выбирается состав системы технических средств, которые будут производиться для выполнения работ и возмещения средств инвестора. На нижнем уровне инвестор стремится максимизировать суммарную эффективность выбранных им изделий, исходя из величины своего кредита. В таких условиях выбор системы для выполнения работ и выбор изделий для погашения кредита становятся взаимозависимыми, что приводит к двухуровневым математическим моделям. Задача состоит в том, чтобы найти такой состав системы, который позволил бы рассчитаться с инвестором и выполнить все работы с минимальными суммарными затратами.

Пусть, как и прежде, множество $I = \{1, \ldots, m\}$ задает исходный ряд образцов. Область их применения задается множеством работ $J = \{1, \ldots, n\}$. Для каждого $i \in I$ известно разбиение множества J на подмножества $J_i^1, \ldots, J_i^{T_i}$. Работы из одного подмножества выполняются последовательно. Работы из разных подмножеств выполняются параллельно. Для каждого $i \in I$ введем следующие обозначения:

$\beta_i \geq 0$ — стоимость одного изделия для инвестора;

$\alpha_i \geq 0$ — оценка эффективности одного изделия инвестором.

Зададим переменные задачи:

$$x_{ij} = \begin{cases} 1, & \text{если } j\text{-я работа выполняется изделиями } i\text{-го образца,} \\ 0 & \text{в противном случае,} \end{cases}$$

$v_i \geq 0$ — объем производства изделий i-го образца,

$w_i \geq 0$ — число изделий i-го образца, проданных инвестору.

Через $g_i(v_i) \geq 0$ обозначим затраты на разработку и производство i-го изделия в зависимости от объема производства $v_i \geq 0$. Будем предпола-

гать, что это кусочно-линейные вогнутые неубывающие функции. Через $B \geq 0$ обозначим величину кредита. С использованием введенных обозначений получаем следующую задачу двухуровневого программирования: найти

$$\min_{v,y} \sum_{i \in I} \{g_i(v_i) + \sum_{j \in J} c_{ij} x_{ij}\}$$

при условиях:

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

$$\sum_{t=1}^{T_i} \max_{j \in J_i^t} p_{ij} x_{ij} \leq v_i - w_i^*(v, x),\ i \in I,$$

$$v_i \geq 0, x_{ij} \in \{0, 1\},\ i \in I, j \in J,$$

где $w^*(v, x)$ — оптимальное решение задачи инвестора, имеющей вид линейной задачи о ранце ЛР(v): найти

$$\max_{w} \sum_{i \in I} \alpha_i w_i$$

при условиях:

$$\sum_{i \in I} \beta_i w_i \leq B,$$

$$0 \leq w_i \leq v_i,\ i \in I.$$

Целевая функция выражает суммарные затраты на разработку, производство и эксплуатацию изделий. Первое ограничение гарантируют выполнение всех работ области применения. Второе ограничение позволяет выполнять работы только имеющимися в наличии изделиями. Целевая функция внутренней задачи задает эффективность изделий, выбранных инвестором. Первое неравенство в задаче ЛР(v) ограничивает финансовые возможности инвестора величиной его кредита. Второе неравенство позволяют инвестору закупать изделия в количествах, не превышающих объемы их производства. Обозначим полученную задачу двухуровневого программирования через ВР.

При $B = 0$ в задаче ЛР(v) имеется единственное оптимальное решение $w^* = 0$. В этом случае получаем задачу выбора ряда изделий [10]:

$$\min_{v,y} \sum_{i \in I} \{g_i(v_i) + \sum_{j \in J} c_{ij} x_{ij}\},$$

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

$$\sum_{t=1}^{T_i} \max_{j\in J_i^t} p_{ij}x_{ij} \le v_i, i \in I,$$

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

Пусть величины $\alpha_i/\beta_i, i \in I$, попарно различны и множество I упорядочено так, что $\alpha_1/\beta_1 > \alpha_2/\beta_2 > \ldots > \alpha_m/\beta_m > 0$. В этом случае задача ЛР(v) имеет единственное оптимальное решение. Следовательно, задача ВР невырождена. Поэтому каждое ее оптимальное решение является наилучшим гарантированным решением [41]. Вырожденный случай рассматривается аналогично, но требует более громоздких выкладок [42].

Лемма 1 *Существует оптимальное решение задачи ВР с единственной ненулевой компонентой вектора w.*

Доказательство. Пусть $(v_i^*), (x_{ij}^*), (w_i^*)$ — произвольное оптимальное решение задачи ВР. Тогда найдется такой номер s, что

$$\sum_{i=1}^{s} \beta_i w_i^* = B, \quad w_i^* = v_i^* \text{ при } i < s \text{ и } 0 < w_s^* \le v_s^*.$$

Рассмотрим вспомогательную задачу: найти

$$\min_{v\ge 0} \sum_{1\le i\le s-1} g_i(v_i) + g_s(v_s^* - w_s^* + v_s)$$

при условии

$$\sum_{1\le i\le s} \beta_i v_i = B.$$

Обозначим через $p(B)$ оптимальное значение ее целевой функции. Из предположения о вогнутости и неубывании функции $g_i(v_i)$ следует равенство

$$p(B) = \min\Big\{ \min\{g_i(B/\beta_i) \mid 1 \le i \le s-1\}, g_s(B/\beta_s + v_s^* - w_s^*)\Big\}.$$

Заметим, что вектор

$$\bar{v}_i = \begin{cases} v_i^*, & \text{если } 1 \le i \le s-1, \\ w_s^*, & \text{если } i = s, \end{cases}$$

является допустимым решением вспомогательной задачи. Следовательно, $p(B) \le \sum\limits_{1\le i\le s} g_i(\bar v_i)$. Тогда величина

$$p(B) + \sum_{i=s+1}^{m} \Big\{ g_i(v_i^*) + \sum_{j\in J} c_{ij} x_{ij}^* \Big\} + \sum_{j\in J} c_{sj} x_{sj}^*$$

является нижней оценкой для оптимального значения задачи ВР. Покажем, что эта оценка достигается на допустимом решении этой задачи с единственной ненулевой компонентой вектора w.

Возможны два случая:

1) $p(B) = g_k(B/\beta_k)$ для некоторого $k, 1 \le k \le s$;

2) $p(B) = g_s(v_s^* - w_s^* + B/\beta_s)$.

В первом случае оценка достигается при

$$\bar v_i = \begin{cases} 0, & \text{если } i < s, i \ne k, \\ B/\beta_i, & \text{если } i = k, \\ v_i^* - w_i^*, & \text{если } i = s, \\ v_i^*, & \text{если } i > s, \end{cases} \qquad \bar w_i = \begin{cases} \bar v_i, & \text{если } i < s, \\ 0, & \text{если } i \ge s, \end{cases} \qquad \bar x = x^*,$$

а во втором случае — при

$$\bar v_i = \begin{cases} 0, & \text{если } i < s, \\ B/\beta_i + v_i^* - w_i^*, & \text{если } i = s, \\ v_i^*, & \text{если } i > s, \end{cases} \quad \bar w_i = \begin{cases} 0, & \text{если } i \ne s, \\ B/\beta_i, & \text{если } i = s, \end{cases} \qquad \bar x = x^*.$$

□

Пусть $I_k = \{k, \dots, m\}$, где $1 \le k \le m$.

Теорема 4 *Решение задачи ВР сводится к решению m задач выбора ряда изделий.*

Доказательство. Из леммы 1 следует, что в задаче ВР можно ограничиться поиском оптимальных решений (v^*, x^*, w^*) с одной ненулевой компонентой $w_k^*, k \in I$, такой что $\beta_k w_k^* = B$ и $w_k^* \le v_k^*$.

Пусть (v, x) — произвольное допустимое решение следующей задачи (обозначим ее через ВР_k): найти

$$Z_k^* = \min_{v,x} \sum_{i\in I_{k+1}} \Big\{ g_i(v_i) + \sum_{j\in J} c_{ij} x_{ij} \Big\} + \sum_{j\in J} c_{kj} x_{kj} + g_k(B/\beta_k + v_k$$

при условиях:

$$\sum_{i \in I_k} x_{ij} = 1, j \in J,$$

$$\sum_{t=1}^{T_i} \max_{j \in J_i^t} p_{ij} x_{ij} \leq v_i, i \in I_k,$$

$$v_i \geq 0, x_{ij} \in \{0,1\}, i \in I_k, j \in J.$$

Тогда решение

$$\bar{v}_i = \begin{cases} 0, & \text{если } i < k, \\ B/\beta_i + v_i, & \text{если } i = k, \\ v_i, & \text{если } i \in I_{k+1}, \end{cases}$$

$$\bar{x}_{ij} = \begin{cases} 0, & \text{если } i < k, j \in J, \\ x_{ij}, & \text{если } i \in I_k, j \in J, \end{cases} \qquad \bar{w}_i = \begin{cases} B/\beta_i, & \text{если } i = k, \\ 0, & \text{в противном случае,} \end{cases}$$

является допустимым решением задачи BP с одной ненулевой компонентой вектора w и значения целевых функций на этих решениях совпадают.

Осталось заметить, что для любого k задача BP_k является задачей выбора ряда изделий. □

Если функции $g_i(v_i)$ имеют вид:

$$g_i(v_i) = \begin{cases} 0, & \text{если } v_i = 0, \\ f_i + c_i v_i, & \text{если } v_i > 0, \end{cases}$$

то получаем сведение задачи BP к серии из m простейших задач размещения. В работах [42, 76] это сведение используется для построения точных алгоритмов и приближенных алгоритмов с гарантированными оценками точности. Более того, оно позволяет получать достаточные условия, при выполнении которых задача BP решается эффективно.

Глава 2

Лагранжевы релаксации

В начале семидесятых годов прошлого столетия была выдвинута одна, как оказалось, очень продуктивная идея для решения NP-трудных задач. Многие из них позволяют разбить систему ограничений задачи на две группы так, что удаление одной из них превращает задачу в полиномиально разрешимую. Эти ограничения заносятся в целевую функцию с некоторыми коэффициентами, множителями Лагранжа. Релаксированная таким способом задача дает оценку снизу на оптимум исходной задачи, если решается задача на минимум, и оценку сверху для задач на максимум. Кроме того, оптимальное решение релаксированной задачи локальными изменениями часто удается перестроить в допустимое решение исходной задачи, что приводит к приближенным алгоритмам с апостериорной оценкой точности.

В данной главе идея Лагранжевых релаксаций будет использована для приближенного решения задач размещения и унификации. В первом разделе приводится общая схема Лагранжевых релаксаций, их геометрическая интерпретация и методы субградиентной оптимизации для поиска оптимальных множителей. Во втором разделе эта схема применяется к задаче выбора оптимального состава системы технических средств при многоэтапном процессе выполнения работ. В третьем разделе предложенный подход обобщается на случай двухуровневых систем. В четвертом разделе те же идеи применяются к динамическим задачам унификации и стандартизации. Каждый раз решение релаксированных задач представляет определенную трудность, связанную как с выбором наиболее сильной формулировки ЦЛП, так и с построением точного решения задачи. Такое решение позволяет вычислить субградиент максимизируемой функции и применить известные схемы негладкой оптимизации для решения двойственной задачи. Практические результаты такого подхода

иллюстрируются численными экспериментами, подтверждающими работоспособность основной идеи и указывающими границы ее применения.

2.1 Общая схема

Рассмотрим задачу P целочисленного линейного программирования в следующей общей форме:

$$\min_x\{cx \mid Ax \le b, Ex \le d, x \in \mathbb{N}^n\},$$

где ограничения $Ex \le d$ будем считать *простыми*, если задача P становится полиномиально разрешимой после удаления *сложных* ограничений $Ax \le b$. Другими словами, ограничения $Ax \le b$ нарушают структуру ограничений $Ex \le d$ и не позволяют эффективно ее использовать. Например, после удаления сложных ограничений задача может распадаться на независимые подзадачи, легко решаемые уже известными алгоритмами.

Лагранжевой релаксацией задачи P относительно ограничений $Ax \le b$ называют задачу P_λ с заданными неотрицательными множителями Лагранжа $\lambda \ge 0$, т.е. задачу вида:

$$\min_x\{cx + \lambda(b - Ax) \mid Ex \le d, x \in \mathbb{N}^n\}.$$

Так как область допустимых решений расширилась, сложные ограничения удалены, то в целевую функцию добавляется штраф за их нарушение. Основная цель — получить задачу P_λ, решать которую было бы легче, чем исходную, релаксируя при этом как можно меньше ограничений.

Обозначим через m число множителей Лагранжа (число сложных ограничений), а через $v(\cdot)$ — оптимальное значение оптимизационной задачи. Известно [127], что при любом выборе вектора $\lambda \in \mathbb{R}^m_+$ величина $v(P_\lambda)$ является нижней оценкой для $v(P)$, т.е. $v(P_\lambda) \le v(P)$. Поиск наилучшей такой оценки приводит к двойственной задаче D: найти

$$\max_\lambda\{v(P_\lambda) \mid \lambda \in \mathbb{R}^m_+\}$$

относительно ограничений $Ax \le b$.

2.1.1 Методы субградиентной оптимизации

Среди численных методов решения двойственной задачи наибольшее распространение получили методы субградиентной оптимизации. Это итеративные процедуры, формирующие последовательность неотрицательных векторов $\{\lambda^k\}$, начиная с некоторого начального значения λ^0, по следующему правилу:

$$\lambda^{k+1} = \max\{0, \lambda^k + t^k(b - Ax^k)\},$$

где x^k — оптимальное решение задачи P_{λ^k}, а t^k — размер шага в направлении субградиента $(b - Ax^k)$ функции $\rho(\lambda) = v(P_\lambda)$ в точке $\lambda = \lambda^k$. Известно [183], что $v(P_{\lambda^k}) \to v(D)$ при $t^k \to 0$ и $\sum_{k=0}^{\infty} t^k \to \infty$. Однако на практике используют другие правила выбора шага, например [146]:

$$t^k = \frac{\Theta^k(F - v(P_{\lambda^k}))}{||b - Ax^k||^2},$$

где Θ^k — скаляр, $0 < \Theta^k \leq 2$ и F — верхняя оценка на $v(D)$. В качестве F обычно берется значение целевой функции задачи P на некотором допустимом решении. Последовательность Θ^k начинается, как правило, с $\Theta^0 = 2$ и через фиксированное число итераций, зависящее от размерности задачи, эта величина делится пополам.

Среди методов субградиентной оптимизации следует отметить методы Н.З. Шора [192] с растяжением пространства и движением в направлении разности двух последовательных субградиентов, а также "объемные"алгоритмы (the volume algorithms [84, 87]). Эти модификации позволяют сглаживать недостатки исходного алгоритма субградиентной оптимизации и быстрее получать оптимальное решение двойственной задачи.

Если на некотором шаге невязка $b - Ax^k$ оказалась равной нулю, то x^k — оптимальное решение исходной задачи. Такое равенство влечет отсутствие разрыва двойственности, т.е. $v(D) = v(P)$, что встречается редко. Чаще всего невязка не равна нулю для оптимальных множителей Лагранжа, но если она достаточно мала, то получается оптимальное решение *близкой* задачи, где вектор b заменен на Ax^k. Конечно, решение x^k может не быть допустимым в исходной задаче, но его часто удается достроить до допустимого локальными изменениями. В этом случае получается приближенный алгоритм с апостериорной оценкой точности.

Такой подход к решению задач дискретной оптимизации предложил использовать H. Everett [110]. Однако бум в этой области начался после

выхода в свет работы [145], посвященной задаче коммивояжера. Именно Лагранжевы релаксации позволили получить рекордные нижние оценки для этой задачи. Успех в задаче коммивояжера послужил толчком для теоретических и прикладных исследований. Для многих трудных оптимизационных задач, особенно задач большой размерности, удалось на этом пути получить работоспособные алгоритмы, верхние и нижние оценки оптимума, которые потом использовались как в точных схемах так и в метаэвристиках. Библиография в этой области насчитывает сотни статей. Обзор результатов можно найти в [125, 133, 166].

2.1.2 Геометрическая интерпретация

Известно [127], что двойственная задача D эквивалентна следующей задаче линейного программирования $\widetilde{P}$: найти

$$\min_x\{cx \mid Ax \le b,\ x \in conv(X)\},$$

где $X = \{x \in \mathbb{N}^n \mid Ex \le d\}$ — множество допустимых решений Лагранжевой релаксации P_λ и $conv(\cdot)$ — выпуклая оболочка, т.е. $v(D) = v(\widetilde{P})$. Отсюда, в частности, следует, что для линейной релаксации $\overline{P}$, получающейся переходом от целочисленных переменных $x \in \mathbb{N}^n$ к непрерывным $x \ge 0$, Лагранжева двойственная задача $\overline{D}$ относительно ограничений $Ax \le b$, т.е. задача вида

$$\max_\lambda v(\overline{P}_\lambda) = \max_\lambda \min_x\{cx + \lambda(b - Ax) \mid Ex \le d, \quad x \ge 0\},$$

не имеет разрыва двойственности, $v(\overline{D}) = v(\overline{P})$.

Таким образом, $v(P) \ge v(D) \ge v(\overline{P})$ и оптимальные значения двойственных переменных задачи линейного программирования $\overline{P}$ дают хорошее приближение для оптимальных значений множителей Лагранжа задачи D.

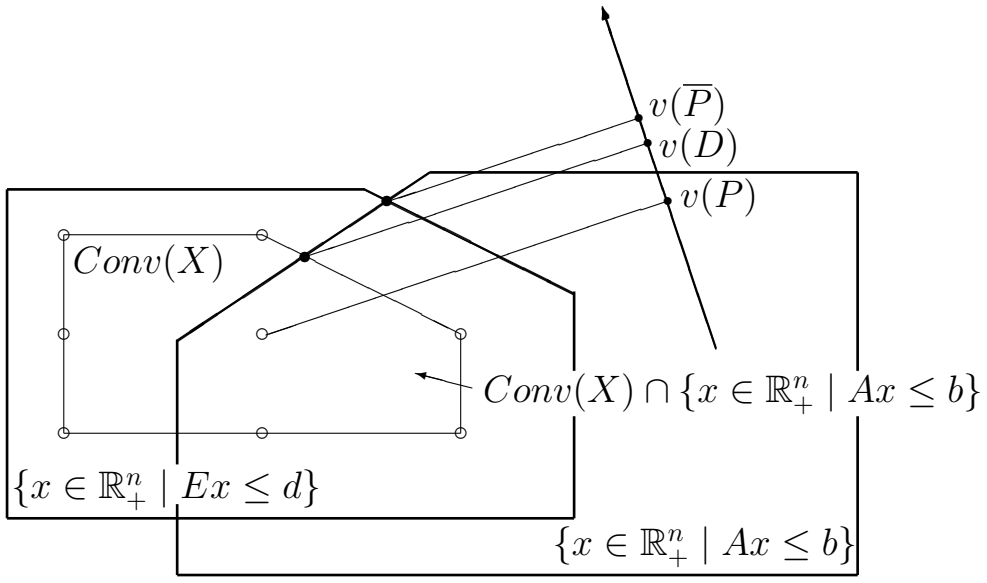


Рис. 1.

Кроме того, если система ограничений $Ex \leq d$ обладает свойством целочисленности, т.е. $v(P_\lambda) = v(\overline{P}_\lambda)$ для каждого $\lambda \in R^m_+$, то $v(D) = v(\widetilde{P}) = v(\overline{P})$. Свойство целочисленности системы влечет целочисленность любой вершины многогранника задачи $\overline{P}_\lambda$, и задача P_λ оказывается не сложнее задачи линейного программирования. Следовательно, для того чтобы Лагранжева релаксация давала лучшую нижнюю оценку, чем линейная релаксация $\overline{P}$, она должна быть *сложнее*, чем $\overline{P}$, и система ограничений $Ex \leq d$ не должна давать точного описания многогранника $conv(X)$ (см. рис. 1).

Указанное свойство может оказаться полезным при выборе системы ограничений для релаксации. Интуитивно понятно, что разбивая разными способами ограничения задачи на две части, можно получать разные двойственные задачи и, как следствие, разные нижние оценки. Например, рассмотрим Лагранжеву двойственную задачу D' относительно ограничений $Ex \leq d$:

$$\max_{\beta} \Big\{ \min_{x} \{cx + \beta(d - Ex) \mid x \in X'\} \Big\},$$

где $X' = \{x \in \mathbb{N}^n \mid Ax \leq b\}$. Если обе системы $Ax \leq b$ и $Ex \leq d$ обладают свойством целочисленности, то $v(D) = v(D') = v(\overline{P})$. Если же этим свойством обладает только одна система, скажем, $Ex \leq d$, а у второй системы нет этого свойства, то $v(D') \geq v(D) = v(\overline{P})$ и неравенство может

быть строгим. В последующих разделах это свойство будет использовано для получения нижних оценок в задачах размещения и унификации.

2.1.3 Лагранжева декомпозиция

Многие практические задачи имеют сложную структуру ограничений. Любое разбиение их на две части может оказаться неэффективным с точки зрения Лагранжевых релаксаций. Задачу не удается разбить на части, релаксированная задача не распадается на подзадачи и т.п. В этом случае часто оказывается полезным прием, получивший название Лагранжевой деомпозиции [134]. Представим исходную задачу P в следующем эквивалентном виде: найти

$$\min_{x,x'}\{c(x+x')/2 \mid Ax' \le b,\ Ex \le d,\ x = x',\ x, x' \in \mathbb{N}^n\}.$$

Тогда двойственная задача D'' относительно искусственно введенных ограничений $x = x'$ имеет вид:

$$\max_{\gamma}\Big\{\min_{x}\{(c/2-\gamma)x \mid x \in X\} + \min_{x'}\{(c/2+\gamma)x' \mid x' \in X'\}\Big\},$$

и релаксированная задача оказалась разбитой на две части. Более того, для задачи D'' справедливо равенство

$$v(D'') = \min_{x}\{cx \mid x \in conv(X) \cap conv(X')\},$$

откуда следует неравенство

$$v(D'') \ge \max\{v(D), v(D')\} \ge v(\overline{P}).$$

Другими словами, Лагранжева декомпозиция гарантирует нижнюю оценку оптимума не хуже, чем двойственные задачи относительно ограничений $Ax \le b$ и $Ex \le d$ в отдельности. Конечно, $v(D'') = \max\{v(D), v(D')\}$, если хотя бы одна из систем ограничений обладает свойством целочисленности, и $v(D'') = v(\overline{P})$, если это свойство имеется у обеих систем. Если же обе системы не обладают указанным свойством, то неравенство может оказаться строгим.

Приведенные свойства Лагранжевой декомпозиции следуют из равенства $v(D) = v(\widetilde{P})$. Однако с практической точки зрения более важным может оказаться тот факт, что задачи D и $\widetilde{P}$ являются линейными и взаимодвойственными. Для того, чтобы доказать оптимальность вектора λ для задачи D потребуется предъявить и оптимальное решение $\widetilde{x}$ для

задачи $\widetilde{P}$ в качестве сертификата. Таким образом, алгоритмы решения двойственной задачи фактически решают и задачу $\widetilde{P}$, что может быть использовано для приближенного решения исходной задачи.

2.2 Лагранжевы релаксации для задачи ВСС

Идеи Лагранжевых релаксаций широко применялись для решения дискретных задач размещения [47, 48, 92, 100, 101]. Для задач стандартизации и унификации этот подход впервые был опробован в работе [37]. Позже аналогичные идеи использовались при решении динамических задач с ограничениями на объемы производства и временем жизни изделий [39], обобщенной задачи о назначении [52, 211] и др. В данном разделе будет показано, как с помощью Лагранжевых релаксаций можно получать верхние и нижние оценки оптимума в задаче ВСС из раздела 1.5.1.

2.2.1 Сильные и слабые формулировки

Напомним, что задача ВСС состоит в поиске состава системы технических средств, который позволяет выполнить заданные работы с минимальными суммарными затратами. Процесс выполнения работ разбит на этапы. Сначала выполняются работы из множества $J_1 \subset J$, затем из множества $J_2 \subset J$ и т.д. Математическая постановка задачи в терминах частично–целочисленного программирования, представленная в разделе 1.5.1, имеет следующий вид: найти

$$\min \sum_{i \in I} (c_i^0 x_i + c_i v_i + \sum_{j \in J} c_{ij} x_{ij}) \tag{2.1}$$

при условиях:

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{2.2}$$

$$\sum_{i \in I} x_i \leq p, \tag{2.3}$$

$$\sum_{j \in J_l} p_{ij} x_{ij} \leq v_i^0 + v_i - \sum_{\tau=1}^{l-1} s_{i\tau} \sum_{j \in J_\tau} p_{ij} x_{ij}, \quad i \in I, l = 1, \ldots, L, \tag{2.4}$$

$$0 \leq v_i \leq V_i, \quad i \in I, \tag{2.5}$$

$$0 \leq x_{ij} \leq x_i, \quad i \in I, j \in J, \tag{2.6}$$

$$x_i \in \{0,1\}, \quad i \in I. \tag{2.7}$$

Эту формулировку задачи обозначим через P. Для получения нижних оценок в задачах частично–целочисленного программирования традиционно используется линейная релаксация, при которой условие булевости переменных заменяется на требование их принадлежности единичному отрезку

$$0 \le x_i \le 1, \quad i \in I. \tag{2.8}$$

Обозначим такую формулировку задачи через $\overline{P}$, а значение ее целевой функции через $v(\overline{P})$. Очевидно, что $v(\overline{P}) \le v(P)$. Задача $\overline{P}$ является задачей линейного программирования, и значение $v(\overline{P})$ может быть найдено за полиномиальное время. Значение $v(\overline{P})$, как правило, не совпадает с $v(P)$, и величина разности $v(P) - v(\overline{P})$ может достигать значительных размеров [163].

Используя различные эквивалентные переформулировки задачи, можно улучшить оценку $v(\overline{P})$. Заменим в исходной задаче ограничения (2.4), (2.5) на условия

$$\sum_{j \in J_l} p_{ij} x_{ij} \le v_i^0 x_i + v_i - \sum_{\tau=1}^{l-1} s_{i\tau} \sum_{j \in J_\tau} p_{ij} x_{ij}, \quad i \in I, \quad l = 1, \ldots, L, \tag{2.9}$$

$$0 \le v_i \le V_i x_i, \quad i \in I. \tag{2.10}$$

Обозначим полученную задачу через P_x. Задачи P и P_x эквивалентны, но $v(\overline{P}) \le v(\overline{P}_x)$, так как область допустимых значений в задаче $\overline{P}_x$ не шире, чем в задаче $\overline{P}$. Можно показать [51], что для любого $N > 0$ существует семейство исходных данных, на котором одновременно $v(\overline{P}_x) > N v(\overline{P})$ и $v(P) > N v(\overline{P}_x)$.

2.2.2 Нижние оценки оптимума

Рассмотрим две релаксации задачи, при которых в целевую функцию заносятся ограничения (2.2) и (2.9), (2.10) соответственно. В первом случае получаем задачу $LR_1(\lambda)$: найти

$$\min \sum_{i \in I} \left(c_i^0 x_i + c_i v_i - \sum_{j \in J} (\lambda_j - c_{ij}) x_{ij} \right) + \sum_{j \in J} \lambda_j$$

при условиях: (2.3), (2.9), (2.10), (2.6), (2.7).

Во втором случае — задачу $LR_2(\varphi, \psi)$: найти

$$\min \sum_{i \in I} \Big\{ \big(c_i^0 - \sum_{l=1}^{L} v_i^0 \varphi_{il} - \psi_i V_i\big) x_i + \big(c_i - \sum_{l=1}^{L} \varphi_{il} + \psi_i\big) V_i$$

$$+ \sum_{l-1}^{L} \sum_{j \in J} \big(c_{ij} + p_{ij}(\varphi_{il} + \sum_{t=l+1}^{L} s_{it} \varphi_{it})\big) x_{it} \Big\}$$

при условиях: (2.2), (2.3), (2.6), (2.7).

При любом выборе неотрицательных множителей Лагранжа значения $v(LR_1(\lambda))$ и $v(LR_2(\varphi, \psi))$ являются нижними оценками для величины $v(P)$. Решения двойственных задач D и C:

$$v(D) = \max_{\lambda \geq 0} v(LR_1(\lambda)),$$

$$v(C) = \max_{\varphi \geq 0,\, \psi \geq 0} v(LR_2(\varphi, \psi))$$

позволяют получать наилучшие из таких оценок.

Теорема 5 *Справедливы соотношения* $v(\overline{P}) \leq v(\overline{P}_x) = v(D) \leq v(C) \leq v(P)$.

Доказательство. Первое и последнее неравенства очевидны. Убедимся в том, что $v(\overline{P}_x) = v(D)$. Из соотношений двойственности для линейного программирования получаем $v(\overline{P}_x) = v(\overline{D})$. Остается проверить, что $v(\overline{D}) = v(D)$. Покажем, что $v(\overline{LR}_1(\lambda)) \geq v(LR_1(\lambda))$, откуда и будет следовать требуемое равенство. Обозначим через $a_i(\lambda)$ оптимальное значение целевой функции в задаче:

$$a_i(\lambda) = \max \Big\{ \sum_{j \in J} (\lambda_j - c_{ij}) x_{ij} - c_i v_i - c_i^0 \Big\}$$

при условиях: (2.4), (2.5) и $0 \leq x_{ij} \leq 1$, $j \in J, i \in I$,

а через $v(RN(a))$ — оптимальное значение целевой функции в простейшей задаче о ранце:

$$v(RN(a)) = \max \sum_{i \in I} a_i(\lambda) x_i,$$

$$\sum_{i\in I} x_i \le p,$$

$$x_i \in \{0,1\}, \quad i \in I.$$

Тогда $v(LR_1(\lambda)) = \sum\limits_{j\in J}\lambda_j - v(RN(a))$.

Пусть $(x_i^*), (v_i^*), (x_{ij}^*)$ —оптимальное решение задачи $\overline{LR}_1(\lambda)$. Построим по данному решению допустимое решение $(\widehat{x}_i), (\widehat{v}_i), (\widehat{x}_{ij})$ в задаче $LR_1(\lambda)$. Определим величины $\widehat{a}_i(\lambda)$ неравенствами

$$\widehat{a}_i(\lambda) = \sum_{j\in J}(\lambda_j - c_{ij})x_{ij} - c_i v_i - c_i^0, \quad i \in I,$$

где

$$v_i = \begin{cases} 0, & \text{если } x_i^* = 0, \\ v_i^*/x_i^*, & \text{если } x_i^* > 0, \end{cases} \qquad x_{ij} = \begin{cases} 0, & \text{если } x_i^* = 0, \\ x_{ij}^*/x_i^*, & \text{если } x_i^* > 0. \end{cases}$$

Возьмем в качестве $\widehat{x}_i$ оптимальное решение задачи $RN(\widehat{a})$ и положим $\widehat{v}_i = \widehat{x}_i v_i$, $\widehat{x}_{ij} = \widehat{x}_i x_{ij}$, $i \in I$, $j \in J$. Легко проверить, что построенное решение допустимо в задаче $LR_1(\lambda)$ и, кроме того,

$$\sum_{j\in J}\lambda_j - v(\overline{LR}_1(\lambda)) = \sum_{i\in I}\Big\{\sum_{j\in J}(\lambda_j - c_{ij})x_{ij}^* - c_i v_i^* - c_i^0 x_i^*\Big\} = \sum_{i\in I}\widehat{a}_i(\lambda)x_i^*.$$

Так как $\sum\limits_{i\in I} x_i^* \le k$ и $0 \le x_i^* \le 1$, то x_i^* — допустимое решение задачи $\overline{RN}(\widehat{a})$, но $v\overline{RN}(\widehat{a}) = v(RN(\widehat{a}))$ при целых k, откуда $\sum\limits_{i\in I}\widehat{a}_i(\lambda)\widehat{x}_i \ge \sum\limits_{i\in I}\widehat{a}_i(\lambda)x_i^*$ и $v(\overline{LR}_1(\lambda)) \ge v(LR_1(\lambda))$.

Покажем, что $v(D) \le v(C)$. Так как $v(D) = v(\overline{P}_x)$, а для задач линейного программирования $\overline{P}_x$ и $\overline{C}$ справедливо равенство $v(\overline{P}_x) = v(\overline{C})$, то $v(D) = v(\overline{P}_x) = v(\overline{C}) \le v(C)$. □

Следствие 1 *Задача D разрешима за время, полиномиальное от длины записи исходных данных.*

Если в формулировке задачи $LR_1(\lambda)$ вместо условий (2.9), (2.10) использовать ограничения (2.4), (2.5), то полученная таким образом задача $LR_0(\lambda)$ будет эквивалентна задаче $LR_1(\lambda)$. Соответствующие двойственные задачи D_0 и D также будут эквивалентны, и утверждение теоремы

5 можно сформулировать в виде: $v(\overline{P}) \le v(\overline{P}_x) = v(D_0) \le v(C) \le v(P)$.

Из доказательства теоремы 5 следует, что за полиномиальное время можно не только вычислить величину $v(D)$, но и найти множители Лагранжа λ^*, для которых $v(D) = v(LR_1(\lambda^*))$. Пусть μ — оптимальные двойственные оценки, соответствующие в задаче $\overline{P}_x$ равенствам (2.2). Тогда $v(\overline{P}_x) = v(\overline{LR}_1(\mu)) \le v(D)$, но $v(\overline{P}_x) = v(D)$, откуда $\lambda^* = \mu$.

Решение двойственных задач C и D, как правило, осуществляется методами субградиентной оптимизации. Решение задачи $LR_1(\lambda)$, как следует из доказательства теоремы 5, сводится, по сути, к вычислению величин $a_i(\lambda), i \in I$, что равносильно решению задачи линейного программирования. Задача $LR_2(\varphi, \psi)$ после очевидных преобразований сводится к известной NP-трудной задаче размещения, для решения которой приходится использовать вычислительные схемы типа ветвей и границ.

С точки зрения теории вычислительной сложности задачи C и P одинаково трудны, но практически задачу C решать проще и, поскольку при фиксированных булевых переменных (x_i) значения непрерывных переменных в задаче $LR_2(\varphi, \psi)$ определяются за линейное время, а в задаче P для этого приходится решать задачу линейного программирования, для которой неизвестно строго полиномиального алгоритма. Кроме того, для задачи $LR_2(\varphi, \psi)$ уже создан алгоритм ветвей и границ [10], хорошо работающий на средних размерностях $(n, m \le 100)$, в то время как для исходной задачи создание такого алгоритма связано с определенными трудностями.

2.2.3 Алгоритм решения задачи $LR_1(\lambda)$

Приведем строго полиномиальный алгоритм решения задачи $LR_1(\lambda)$. Этот алгоритм будет прямо следовать из конструктивного доказательства следующего утверждения.

Теорема 6 *При фиксированных множителях $\lambda_j, j \in J$, оптимальное решение задачи $LR_1(\lambda)$ может быть найдено с трудоемкостью $O(m \log m + mn(L + \log n))$.*

Доказательство теоремы вытекает из нижеследующих лемм. Пусть, как и прежде,

$$a_i(\lambda) = \max \Big\{ \sum_{j \in J} (\lambda_j - c_{ij}) x_{ij} - c_i v_i - c_i^0 \Big\}$$

при условиях (2.4), (2.5) и $0 \le x_{ij} \le 1$, $j \in J, i \in I$.

При заданных значениях $a_i(\lambda), i \in I$ задача $LR_1(\lambda)$, как уже отмечалось, сводится к простейшей задаче о ранце $RN(a)$, которая решается с трудоемкостью $O(m \log m)$. Для доказательства теоремы остается указать алгоритм вычисления величин $a_i(\lambda), i \in I$. Так как значения $a_i(\lambda)$ определяются независимо друг от друга, то индекс i в дальнейшем будем опускать. Определим функции $F_l(v), v > 0, l = 1, \ldots, L$ равенствами:

$$F_l(v) = \max \sum_{t=l}^{L} \sum_{j \in J_t} c_j x_j,$$

при условиях

$$\sum_{j \in J_t} p_j x_j \le v - \sum_{r=l}^{t-1} s_r \sum_{j \in J_r} p_j x_j, \quad t = l, \ldots, L,$$

$$0 \le x_j \le 1, \quad j \in J_t, \quad t = 1, \ldots, L,$$

где $c_j = \lambda_j - c_{ij}$. Представим функцию $a(\lambda)$ в следующем виде:

$$a(\lambda) = \max_v \left\{ F_1(v) - c \max(0; v - v^0) - c^0 \mid 0 \le v \le V + v^0 \right\}.$$

Функции $F_l(v), l = 1, \ldots, L$, являются вогнутыми кусочно–линейными функциями. Ниже будет показано, что они имеют не более $2n$ точек излома. Значит, величина $a(\lambda)$ может быть найдена по известной функции $F_1(v)$ за $O(\log n)$ операций.

Введем функции $G_l(v), v \ge 0$, определяемые равенствами:

$$G_l(v) = \max \sum_{j \in J_l} c_j x_j,$$

при условиях

$$\sum_{j \in J_l} p_j x_j \le v,$$

$$0 \le x_j \le 1, \quad j \in J_l.$$

Для функций $F_l(v), l = 1, \ldots, L$, справедливы рекуррентные соотношения:

$$F_L(v) = G_L(v),$$

$$F_l(v) = \max_{0 \le u \le v} \left(G_l(u) + F_{l+1}(v - s_l u) \right), \quad l = 1, \ldots, L.$$

Эти соотношения легко проверяются подстановкой. Они позволяют для каждого $v \geq 0$ определить значение функции $F_l(v)$ по известным функциям $F_{l+1}(v)$ и $G_l(v)$. Так как параметр v является непрерывным, то вычислить таким способом все значения функции $F_l(v)$ не представляется возможным. Однако специальный вид функций $F_l(v)$, а именно их кусочно–линейность, позволяет задавать функции их значениями в точках излома и тем самым вычислять значение $F_l(v)$ только конечное число раз.

Лемма 2 *Если* $F_l(v^0) = G_l(u^0) + F_{l+1}(v^0 - s_l u^0)$ *и* $T_l = 1 - s_l \neq 1$, *то для достаточно малых* $\delta > 0$ *верно равенство*

$$F_l(v^0 + \delta) = F_l(v^0) + \gamma\delta,$$

$$\gamma = \begin{cases} \alpha, & \text{если} \quad s_l\alpha \geq \beta, \\ \beta + T_l\alpha, & \text{если} \quad s_l\alpha < \beta \ \text{и} \ u^0 = v^0, \\ \beta/s_l, & \text{если} \quad s_l\alpha < \beta \ \text{и} \ u^0 < v^0, \end{cases}$$

где α, β — *правые производные функции* $F_{l+1}(\mu), G_l(u)$ *в точках* $\mu = \mu^0 = v^0 - s_l u^0$, $u = u^0$.

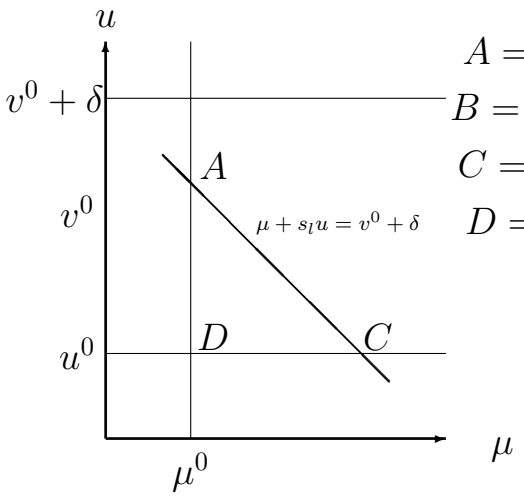
Доказательство. Представим функцию $F_l(v)$ в виде

$$F_l(v) = \max_{u,\mu \geq 0} \big(G_l(u) + F_{l+1}(\mu)\big),$$

$$s_l u + \mu \leq v,$$

$$0 \leq u \leq v,$$

и выделим два случая: $u^0 = v^0$ и $u^0 < v^0$.



$A = (\mu^0, u^0 + \delta/s_l)$,
$B = (\mu^0 + T_l\delta, v^0 + \delta)$,
$C = (\mu^0 + \delta, u^0)$,
$D = (\mu^0, u^0)$,

Рис. 2

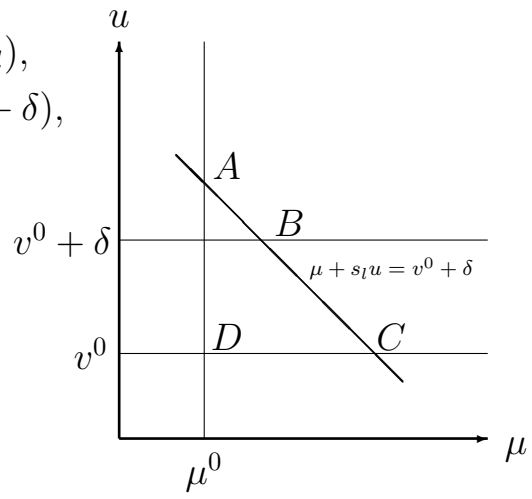


Рис. 3

а) Пусть $u^0 < v^0$ и $F_L(v^0+\gamma) = G_l(u^*) + F_{l+1}(\mu^*)$. Рассмотрим область изменения переменных (μ, u) при увеличении v^0 на δ (см. рис. 2). Очевидно, что точка (μ^*, u^*) лежит на прямой $s_l u + \mu = v^0 + \delta$. В силу вогнутости функций $G_l(u)$ и $F_{L+1}(\mu)$ точка (μ^*, u^*) не выходит из отрезка AC. При достаточно малых $\delta > 0$ функции $G_l(u)$ и $F_{L+1}(\mu)$ можно считать линейными на DA и DC. Сумма двух линейных функций тоже линейна. Максимум линейной функции достигается на границе и можно считать, что (μ^*, u^*) совпадает либо с точкой A, либо с точкой C. Приращение $F_l(v)$ при переходе от точки D к точке C равно

$$\Delta F_C = G_l(u^0) + F_{l+1}(\mu^0+\delta) - G_l(u^0) - F_{l+1}(\mu^0) = \alpha\delta.$$

Аналогично,

$$\Delta F_A = G_l(u^0+\delta/s_l) + F_{l+1}(\mu^0) - G_l(u^0) - F_{l+1}(\mu^0) = \beta\delta.$$

Таким образом, если $\alpha \geq \beta/s_l$, то (μ^*, u^*) совпадает с точкой C и $\gamma = \alpha$. В противном случае $\gamma = \beta/s_l$.

б) Пусть $u^0 = v^0$. В этом случае точка A лежит выше прямой $u = v^0 + \delta$, точка (μ^*, u^*) принадлежит отрезку BC (рис. 3) и

$$\Delta F_B = G_l(v^0+\delta) + F_{l+1}(\mu^0 + T_l\delta) - G_l(v^0) - F_{l+1}(\mu^0) = (\beta + T_l\alpha)\delta. \quad \square$$

Лемма 3 *Функции $F_l(v)$ имеют не более $2n$ точек излома.*

Доказательство. Пусть функции $G_l(u)$ и $F_{l+1}(\mu)$ имеют соответственно n_l и Q_{l+1} точек излома. Будем говорить, что точка излома функции $F_l(v)$ является:

– точкой излома 1-го типа, если для правой производной функции $F_l(v)$ в этой точке верно равенство $\gamma = \alpha$;

– точкой излома 2-го типа, если $\gamma = \beta + T_l\alpha$;

– точкой излома 3-го типа, если $\gamma = \beta/s_l$.

Рассмотрим три случая.

а) Пусть v^0 —точка излома 1-го типа. Тогда $s_l\alpha \geq \beta$,

$$F_l(v^0+\delta) = G_l(u^0) + F_{l+1}(\mu^0+\delta)$$

и следующей точке излома v^1 соответствует точка излома функции $F_{l+1}(v)$. Точка v^1 может быть точкой 1-го или 3-го типа, но не может быть точкой 2-го типа, так как $v^1 = v^0 + \delta > u^0 = u^1$.

б) Пусть v^0 — точка излома 2-го типа. Тогда $u^0 = v^0$, $s_l\alpha < \beta$ и

$$F_l(v^0+\delta) = G_l(u^0+\delta) + F_{l+1}(\mu^0 + T_l\delta).$$

Точка v^1 определяется по точкам излома либо функции $G_l(u)$, либо функции $F_{l+1}(v)$ и может быть точкой излома 1-го или 2-го типа, причем переходу в точку 1-го типа соответствует точка излома функции $G_l(u)$.

в) Пусть v^0 — точка излома 3-го типа. Тогда $u^0 < v^0,\ S_l\alpha < \beta$ и

$$F_l(v^0 + \delta) = G_l(u^0 + \delta/s_l) + F_{l+1}(\mu^0).$$

Точка v^1 определяется либо точкой излома функции $G_l(u)$, либо равенством

$$v^0 + \delta = u^0 + \delta/s_l.$$

В первом случае точка v^1 может быть точкой излома 1-го или 3-го типа, во втором случае — 2-го типа. Отметим, что если раньше точки излома функции $F_l(v)$ определялись точками излома функций $F_{l+1}(v)$ и $G_l(u)$, то в последнем случае появляется дополнительная точка излома, определяемая равенством $v^0+\delta = u^0+\delta/s_l$. Убедимся, что число дополнительных точек излома не превосходит n_l.

Действительно, дополнительная точка излома появляется при переходе от точки 3-го типа к точке 2-го типа. Чтобы снова оказаться в точке 3-го типа, надо пройти через точку 1-го типа, а сделать это из точки 2-го типа можно, только пройдя через точку излома функции $G_l(u)$. Таким образом, число точек излома функции $F_l(v)$ не превосходит $n_l + n_l + Q + l + 1$, откуда

$$Q_1 \le 2n_1 + Q_2 \le 2n_1 + 2n_2 + Q_3 \le \cdots \le 2n.$$

□

Если $s_l = 0$ для некоторого l, то $\gamma = \lambda + \beta$ и дополнительные точки излома не появляются.

Конструктивное доказательство последнего утверждения позволяет построить алгоритм вычисления функции $F_l(v)$. Алгоритм состоит из L шагов, на каждом из которых по уже известным функциям $F_{l+1}(v)$ и $G_l(u)$ строится функция $F_l(v)$. Шаг начинается с построения функции $G_l(u)$. Напомним, что $G_l(u)$ является задачей о ранце с непрерывными переменными (u — объем ранца), точки излома функции $G_l(u)$ определяются с трудоемкостью $O(n_l)$, а построение самой функции $G_l(u)$, $u \ge 0$, требует не более $O(n_l \log n_l)$ операций. По известным значениям $F_{l+1}(v)$ и $G_l(u)$ функция $F_l(v)$ строится за линейное время $O(n_l + Q_{l+1})$, и так как для определения $F_l(v)$ необходимо вычислить все функции

$F_l(v), l = 1, \ldots, L$, то суммарная трудоемкость построения $F_l(v)$ оценивается величиной

$$\sum_{l=1}^{L} O(n_l \log n_l + n_l + Q_{l+1}) \leq O(n \log n + Ln),$$

что и доказывает теорему. □

2.2.4 Построение допустимого решения

Пусть $(x_i^*), (v_i^*), (x_{ij}^*)$ — оптимальное решение задачи $LR_1(\lambda)$ и

$$\Delta_j(\lambda) = \sum_{i \in I} x_{ij}^* - 1, \quad j \in J.$$

Если $\Delta_j(\lambda) = 0$, то есть равенства (2.2) верны для (x_{ij}^*), то $(x_i^*), (v_i^*), (x_{ij}^*)$ — оптимальное решение исходной задачи и $v(LR_1(\lambda))$ совпадает с $v(P)$. Выполнение условий (2.2) означает, что в задаче P_x отсутствует разрыв двойственности. Чаще всего $v(LR_1(\lambda^*)) = v(D) < v(P)$, но величина

$$\delta = \sum_{i \in J} |\Delta_j(\lambda^*)|$$

достаточно мала и решение $(x_i^*), (v_i^*), (x_{ij}^*)$ можно достроить до допустимого решения исходной задачи с помощью процедур координатного спуска. При λ, близких к λ^*, равенства (2.2) нарушаются только для некоторых $j \in J$, и трудоемкость такой процедуры практически оказывается очень низкой. Обозначим через H_D алгоритм, использующий это процедуру в ходе субградиентной оптимизации и выбирающий из полученных решений наилучшее. Близкий по смыслу алгоритм рассматривался в [101].

Обозначим через $P(\Delta)$ задачу, которая отличается от исходной тем, что равенства (2.2) заменены на

$$\sum_{i \in I} x_{ij} = 1 + \Delta_j(\lambda), \quad j \in J.$$

Легко проверить, что оптимальное решение задачи $LR_1(\lambda)$ является оптимальным решением и задачи $P(\Delta)$. Алгоритм H_D просматривает оптимальные решения задач $P(\Delta)$ при разных Δ и достраивает их до допустимого решения задачи P. Если задачи P и $P(\Delta)$ "близки" друг к

другу, то оптимальное решение одной из них является хорошим приближением для другой.

Алгоритм H_D рассматривает различные наборы (x_i^*), что, разумеется, не гарантирует получения оптимального набора задачи P. Более осторожной стратегией, чем в алгоритме H_D, является схема покомпонентного составления набора (x_i), на каждом шаге которой одна из переменных x_i полагается равной 1. Через p шагов алгоритм останавливается, и остальные компоненты вектора полагаются равными нулю. Ниже приводится четыре правила выбора переменной:

1. Положить $x_i = 1$, если $a_i(\lambda^*) = \max(a_k(\lambda^*), k \in I)$.

2. Определить индекс $r \in J$, для которого $\sum\limits_{i\in I} x_{ir}^* = \max_j(\sum\limits_{i\in I} x_{ij}^* \mid j \in J)$, и
положить $x_i = 1$, если $\sum\limits_{j\in J} x_{ij}^* = \max_k(\sum\limits_{j\in J} x_{kj}^* \mid k \in I, x_{kr}^* > 0)$.

3. Положить $x_i = 1$, если $\sum\limits_{j\in J} x_{ij}^* = \max_k \big(\sum\limits_{j\in J} x_{kj}^*, k \in I\big)$.

4. Положить $x_i = 1$, если номер i наиболее часто входил в состав оптимального набора (x_i^*) задачи $LR_1(\lambda)$ в ходе субградиентной оптимизации.

Правило 1 наиболее широко применяется в методах ветвей и границ для задачи размещения с ограничениями на объемы производства [100]. Аналог правила 2 используется в точном алгоритме для задач о покрытии и разбиении [123]. Правило 3 хорошо зарекомендовало себя в простейшей задаче размещения.

Обозначим через H_q алгоритм, использующий q-е правило выбора и делающий не более p шагов. На каждом шаге выбирается переменная x_i и полагается равной 1. Затем производится корректировка множителей Лагранжа и к каждому решению задачи $LR_1(\lambda)$ применяется алгоритм H_D. Результатом работы алгоритма H_q является наилучшее из построенных допустимых решений.

Приведенные алгоритмы базируются на решении задачи $LR_1(\lambda)$, хотя нетрудно построить аналогичные конструкции и для релаксации $LR_2(\varphi, \psi)$. Обозначим через H_C аналог алгоритма H_D, основанный на релаксации $LR_2(\varphi, \psi)$. Понятно, что алгоритм H_C уже не будет иметь полиномиальной трудоемкости, но так как $v(C) \geq v(D)$, то, проигрывая по времени, он может выиграть в точности.

2.2.5 Численные эксперименты

Предложенные алгоритмы тестировались на классе задач следующей размерности: $m = 30,\ n = 30,\ n_l = 5,\ l = 1, \ldots, L,\ L = 6$. Исходные данные формировались датчиком псевдослучайных чисел с равномерным распределением. Параметры задач выбирались из следующих интервалов: $c_i \in [1, 5],\ c_{ij} \in [1, 105],\ p_{ij} \in [1, 15],\ c_i^0 = 100,\ v_i^0 = 2,\ s_{il} = 0, 1$. Матрицы (p_{ij}) и (c_{ij}) на 30% заполнялись числом 10^6. Расчеты проводились при разных значениях параметров p и V ($V_i = V$, см. ограничение (2.5)).

Решения тестовых задач без ограничений (2.3), (2.5) содержат в оптимальном наборе (x_i^*) от 5 до 6 ненулевых элементов, объемы производства меняются от 2 до 11 единиц, и разрыв двойственности в среднем составляет около 3% от $v(C)$.

Наилучшие по точности результаты на исследуемых классах задач при $(p, V) \in \{(5, 30); (3, 1000); (3, 7); (30, 1000)\}$ принадлежат алгоритму H_C. При решении задачи C оптимальные наборы (x_i^*) часто повторяются, число различных наборов в среднем не превосходит трех, а в половине случаев такой набор единственный. Интересно отметить, что при решении задачи D наблюдается совершенно другая картина. Из множества I выделяется группа номеров (от 10 до 15 элементов) с примерно равными значениями $a_i(\lambda)$, к которым эвристики H_q применяют правила выбора. Оптимальные наборы (x_i^*) почти не повторяются, и наилучшее решение эвристики H_q редко совпадает с набором номеров, выбранными по q-му правилу. Другими словами, рекордное значение находится алгоритмом на промежуточном шаге с помощью процедуры H_D. Алгоритмы H_q в среднем дают небольшое уклонение от оптимума на задачах, где оба или одно из ограничений (2.3), (2.5) являются несущественными. Ни одна из эвристик H_q не является доминирующей, и все они могут сильно ошибаться, когда ограничения (2.3), (2.5) одновременно являются "жесткими". Такой класс задач является наиболее трудным для данных эвристик, и на нем целесообразно использовать комбинацию алгоритмов H_D и H_C:

Обозначим через H_{DC} алгоритм, работающий по следующей схеме:

1. Решить задачу D и определить множество

$$I_\varepsilon = \{i \in I \mid a_i(\lambda^*) + \varepsilon \geq \max(a_k(\lambda^*),\ k \in I)\}.$$

2. Решить задачу C на множестве I_ε, используя алгоритм H_C для по-

строения допустимого решения.

Алгоритм H_{DC} на первом шаге формирует множество номеров, с которыми работают эвристики H_q, а на втором шаге вместо правил выбора использует релаксацию LR_2 для построения набора (x_i^*). Такая комбинация релаксаций позволяет использовать достоинства обоих подходов и сглаживает их недостатки. При $\varepsilon = v(D)/100$ алгоритм H_{DC} имеет те же результаты по точности, что и алгоритм H_C, но тратит в несколько раз меньше времени, чем алгоритм H_C.

2.3 Лагранжевы релаксации для задачи $B2CC$

Предположим, что для выполнения работ технические средства объединяются в комплекты. Они могут рассматриваться как новые сложные технические средства, имеющие в своем составе унифицированные составные части. Процесс выполнения работ разбит на этапы. На каждом этапе известная доля комплектов выходит из строя. Задача B2CC состоит в нахождении состава системы комплектов технических средств, который позволяет выполнить все работы с минимальными суммарными затратами.

С помощью обозначений раздела 1.5.2 задача B2CC записывается в следующем виде: найти

$$\min_{x,y,z,v} \Big(\sum_{k \in K} \Big(c_k^{\circ}\ z_k + c_k\ v_k + \sum_{j \in J} c_{kj}\ x_{kj} \Big) + \sum_{i \in I} d_i\ y_i \Big) \tag{2.11}$$

при условиях:

$$\sum_{k \in K} x_{kj} = 1, \quad j \in J, \tag{2.12}$$

$$\sum_{j \in J_l} p_{kj} x_{kj} \le v_k^{\circ} z_k + v_k - \sum_{l'=1}^{l-1} s_{kl'} \sum_{j \in J_{l'}} p_{kj} x_{kj}, \quad k \in K, \quad l = 1, \ldots, L, \tag{2.13}$$

$$0 \le v_k \le V_k z_k, \quad k \in K, \tag{2.14}$$

$$0 \le x_{kj} \le z_k, \quad k \in K, \quad j \in J, \tag{2.15}$$

$$y_i \ge z_k, \quad k \in K_i, \quad i \in I, \tag{2.16}$$

$$y_i, z_k \in \{0, 1\}, \quad i \in I, \quad k \in K. \tag{2.17}$$

Ниже для этой задачи будут предложены две нижние оценки, получаемые в результате ослабления двух различных групп ограничений. Будет показано, что эти оценки несравнимы между собой и могут быть вычислены за полиномиальное время. В ходе численных экспериментов выделены области доминирования одной оценки над другой.

2.3.1 Две нижние оценки

Рассмотрим две вспомогательные задачи L_f и L_h, которые получаются путем внесения в целевую функцию части ограничений задачи. Условиям поставим в соответствие $\sum_{k\in K} x_{kj} = 1, \quad j \in J$, двойственные переменные α_j, $j \in J$. Обозначим через L_f задачу нахождения следующей величины:

$$f(\alpha) = \min_{x,y,z,v} \Big\{ \sum_{k\in K} \Big(c_k^\circ z_k + c_k v_k + \sum_{j\in J} (c_{kj} - \alpha_j) x_{kj} \Big) + \sum_{i\in I} d_i y_i \Big\}$$

при условиях (2.13)–(2.17).

Соответствующая двойственная задача D_f, состоящая в нахождении величины

$$F = \max_{\alpha} \Big\{ f(\alpha) + \sum_{j\in J} \alpha_j \Big\},$$

позволяет вычислять нижнюю оценку целевой функции (2.11). Заменим неравенства (2.16) на условия:

$$y_i \geq \sum_{k\in K_i} x_{kj}, \quad i \in I, \quad j \in J. \tag{2.18}$$

Легко проверить, что при этом область допустимых значений не меняется. Каждому ограничению (2.18) поставим в соответствие двойственные переменные $\beta_{ij} \geq 0$, $i \in I$, $j \in J$, и обозначим через L_h следующую задачу:

$$h(\alpha,\beta) = \min_{x,y,z,v} \Big\{ \sum_{k\in K} \Big(c_k^\circ z_k + c_k v_k + \sum_{j\in J} \Big(c_{kj} - \alpha_j + \sum_{i\in I_k} \beta_{ij} \Big) x_{kj} \Big) + \sum_{i\in I} \Big(d_i - \sum_{j\in J} \beta_{ij} \Big) y_i \Big\}$$

при условиях: (2.13)–(2.15), (2.17).

Решение двойственной задачи D_h, состоящей в нахождении величины

$$H = \max_{\alpha,\beta}\Big\{h(\alpha,\beta) + \sum_{j\in J}\alpha_j\Big\},$$

также доставляет нижнюю оценку для минимального значения целевой функции (2.11).

Теорема 7 *Задачи D_f и D_h полиномиально разрешимы.*

Доказательство. Покажем, что задача D_f полиномиально разрешима. Для каждого $k \in K$ рассмотрим вспомогательную задачу линейного программирования, состоящую в нахождении величины

$$g_k(\alpha) = \max_{x,v}\Big\{\sum_{j\in J}(\alpha_j - c_{kj})x_{kj} - c_k v_k - c_k^{\circ}\Big\}$$

при условиях: (2.13), (2.14) и $0 \le x_{kj} \le 1$, $k \in K$, $j \in J$.

Убедимся в том, что задача $\overline{L}_g$ нахождения величины

$$\overline{f}_g(\alpha) = \min_{y,z}\Big\{\sum_{i\in I} d_i y_i - \sum_{k\in K} g_k(\alpha) z_k\Big\}$$

при условиях: (2.16) и $0 \le y_i \le 1$, $0 \le z_k \le 1$, $i \in I$, $k \in K$

имеет оптимальное целочисленное решение.

Пусть y_i^*, z_k^* — оптимальное решение задачи $\overline{L}_g$. Очевидно, что если $g_k(\alpha) \le 0$, то $z_k^* = 0$. Если же $g_k(\alpha) > 0$, то $z_k^* = \min\limits_{i\in I_k} y_i^*$. Положим $\delta = \min\{y_i^* > 0, i \in I\}$; $I_\delta = \{i \in I \mid y_i^* = \delta\}$; $K_\delta = \{k \in K \mid z_k^* = \delta\}$; $\varepsilon_1 = \sum\limits_{i\in I_\delta} d_i$; $\varepsilon_2 = \sum\limits_{k\in K_\delta} g_k(\alpha)$. Если $\delta = 1$, то решение y_i^*, z_k^* — целочисленное. Предположим, что $\delta < 1$. Если $\varepsilon_1 > \varepsilon_2$, то уменьшение величин y_i^*, z_k^*, $i \in I_\delta$, $k \in K_\delta$, приводит к уменьшению целевой функции. Если же $\varepsilon_1 < \varepsilon_2$, то увеличение значений y_i^*, z_k^*, $i \in I_\delta$, $k \in K_\delta$, приводит к тому же эффекту. Следовательно, $\varepsilon_1 = \varepsilon_2$, но тогда, полагая $y_i = 0$, $z_k = 0$, $i \in I_\delta$, $k \in K_\delta$, получаем новое оптимальное решение с большим значением параметра δ. Таким образом, если в оптимальном решении все дробные компоненты положить равными нулю, то получим оптимальное целочисленное решение задачи $\overline{L}_g$.

Заметим, что задачи $\overline{L}_g$ и $\overline{L}_f$ эквивалентны и оптимальные значения их целевых функций совпадают. Следовательно, задача D_f эквивалентна задаче $\overline{D}_f$, откуда и следует требуемое.

Убедимся в полиномиальной разрешимости задачи D_h. Для каждого $k \in K$ рассмотрим вспомогательную задачу линейного программирования

$$g_k(\alpha,\beta) = \max_{x,v}\Big\{\sum_{j\in J}\Big(\alpha_j - c_{kj} + \sum_{i\in I_k}\beta_{ij}\Big)x_{kj} - c_k v_k - c_k^{\circ}\Big\}$$

при условиях: (2.13), (2.14) и $0 \le x_{kj} \le 1$, $k \in K$, $j \in J$.

Тогда задача L_h состоит в нахождении величины

$$h(\alpha,\beta) = \min_{y,z}\Big\{\sum_{i\in I}\Big(d_i - \sum_{j\in J}\beta_{ij}\Big)\ y_i - \sum_{k\in K} g_k(\alpha,\beta)z_k\Big\}$$

при условии (2.17).

В отличие от задачи L_f, переменные y_i, $i \in I$, уже не связаны с другими переменными и их оптимальные значения определяются только знаком соответствующих коэффициентов. Значит, задача L_h является частным случаем задачи L_f и для нее переход к линейной релаксации также не меняет минимального значения целевой функции. Следовательно, задачи D_h и $\overline{D}_h$ эквивалентны. □

Задачи линейного программирования $\overline{D}_f$ и $\overline{D}_h$ имеют большую размерность, и решение их стандартными методами требует значительных усилий. Для вычисления нижних оценок F и H целесообразно использовать методы субградиентной оптимизации. На каждой итерации этого метода решаются задачи L_f и L_h и по невязке в условиях (2.12) и (2.18) производится корректировка величин α_j, β_{ij}, $i \in I$, $j \in J$. Если задачи L_f и L_h полиномиально разрешимы, то такой подход имеет определенные преимущества. Сложность решения задачи L_h определяется сложностью нахождения величин $g_k(\alpha,\beta)$. Вычисление коэффициентов при переменных x_{kj} требует не более $O(qmn)$ операций. Нахождение величин $g_k(\alpha,\beta)$ может быть осуществлено с использованием не более $O(nq(L+\log q))$ операций, $q = |K|$. Общая трудоемкость решения задачи L_h оценивается сверху величиной $O(qmn + nq(L+\log q))$.

Теорема 8 *Задача L_f может быть решена с трудоемкостью $O(m^2n + nq(L+\log q))$.*

Доказательство. Представим задачу L_f в виде

$$f(\alpha) = \min_{y,z}\Big\{\sum_{i\in I} d_i\ y_i - \sum_{k\in K} g_k(\alpha)z_k\Big\}$$

при условиях: (2.16), (2.17).

Без ограничения общности будем считать, что $g_k(\alpha) \geq 0$, $k \in K$. Запишем условия (2.16) в виде равенства

$$z_k = \prod_{i \in I_k} y_i, \quad k \in K,$$

и исключим переменные z_k из рассмотрения. Тогда задача L_f может быть представлена как задача минимизации полинома

$$\Phi(y) = \sum_{i \in I} d_i \; y_i - \sum_{k \in K} g_k(\alpha) \prod_{i \in I_k} y_i$$

от булевых переменных y_i, $i \in I$. Решение этой задачи может быть получено путем сведения ее к задаче о минимальном разрезе на двудольной сети [188, 88]. Пусть $G = (V, E)$ — двудольная сеть с источником s, стоком t, множеством вершин $V = I \cup K \cup \{s, t\}$ и множеством дуг $E = E_1 \cup E_2 \cup E_3$, где I — множество вершин одной доли, K — множество вершин другой доли, $E_1 = \{(s, i) \mid i \in I\}$, $E_2 = \{(i, k) \mid i \in I_k, k \in K\}$, $E_3 = \{(k, t) \mid k \in K\}$. Каждая дуга e имеет пропускную способность $w(e) \geq 0$, определяемую следующим образом:

$$w(e) = \begin{cases} d_i, & \text{если } e = (s, i) \in E_1, \\ W, & \text{если } e = (i, k) \in E_2, \text{ где } W > \sum\limits_{i \in I} d_i, \\ g_k(\alpha), & \text{если } e = (k, t) \in E_3. \end{cases}$$

Пусть φ — величина минимального разреза сети G. Очевидно, что минимальный разрез не содержит дуг из множества E_2. Ясно, что для любого решения $y_i \in \{0, 1\}$ легко построить разрез сети G с пропускной способностью $\Phi(y) + \sum\limits_{k \in K} g_k(\alpha)$. Для этого достаточно взять все дуги (s, i), для которых $y_i = 1$, и все дуги (k, t), для которых $\prod\limits_{i \in I_k} y_i = 0$. Убедимся в справедливости обратного утверждения, т. е. что для любого минимального разреза сети G существует решение $y_i \in \{0, 1\}$ такое, что

$$\varphi = \Phi(y) + \sum_{k \in K} g_k(\alpha). \tag{2.19}$$

Пусть множество дуг $E' = E'_1 \cup E'_3$ является минимальным разрезом сети G. Положим

$$y_i = \begin{cases} 1, & \text{если } e = (s, i) \in E'_1, \\ 0 & \text{в противном случае} \end{cases}$$

и проверим справедливость равенства (2.19). Для этого достаточно убедиться в том, что множество E'_3 совпадает с множеством дуг $E_y = \{e = (k,t) \mid \prod\limits_{i \in I_k} y_i = 0\}$.

Сначала покажем, что $E'_3 \supseteq E_y$. Предположим, что существует дуга $e \in E_y$ такая, что $e \notin E'_3$, т. е. для некоторого k справедливо равенство $\prod\limits_{i \in I_k} y_i = 0$ и дуга $(k,t) \notin E'_3$. В множестве I_k выберем i таким, что $y_i = 0$. Тогда дуга $(s,i) \notin E'_1$ и путь из s в t состоит из последовательности дуг (s,i), (i,k), (k,t), что невозможно.

Покажем теперь, что $E'_3 \subseteq E_y$. Предположим, что существует дуга $e = (k,t) \in E'_3$ такая, что $e \notin E_y$, т. е. $y_i = 1$, $i \in I_k$. По построению любой путь из s в k проходит только через вершины множества I_k. Так как все ребра (s,i), $i \in I_k$, принадлежат E'_1, то удаление ребра (k,t) оставляет множество $E' \setminus \{(k,t)\}$ разрезом, что противоречит минимальности E'.

Таким образом, решение задачи L_f сводится к нахождению минимального разреза на двудольной сети G, что требует $O(m^2 n)$ операций [135]. Вычисление коэффициентов $g_k(\alpha)$, $k \in K$, можно осуществить за $O(nq(L + \log q))$ операций [37]. Общая трудоемкость решения задачи L_f оценивается сверху величиной $O(m^2 n + nq(L + \log q))$. □

2.3.2 Соотношение величин F и H

Покажем, что нижние оценки F и H не сравнимы между собой. При $v_k^\circ \geq \sum\limits_{j \in J} p_{kj}$, $k \in K$, исходная задача упрощается и состоит в нахождении величины

$$\min_{x,y,z} \Big(\sum_{k \in K} \Big(c_k^\circ \; z_k + \sum_{j \in J} c_{kj} \; x_{kj} \Big) + \sum_{i \in I} d_i \; y_i \Big)$$

при условиях:

$$\sum_{k \in K} x_{kj} = 1, \quad j \in J,$$

$$0 \leq x_{kj} \leq z_k, \quad k \in K, \quad j \in J,$$

$$y_i \geq z_k, \quad k \in K_i, \quad i \in I,$$

$$y_i, z_k \in \{0,1\}, \quad i \in I, \quad k \in K.$$

В работе [88] рассматривался частный случай этой задачи, когда $|I_k| = 2$ для всех $k \in K$. Установлено, что в этом случае $H \geq F$. (Ограничение $|I_k| = 2$, $k \in K$, несущественно, и утверждение справедливо для произвольных множеств I_k.)

Приведем пример исходных данных, на которых справедливо обратное неравенство, точнее, $F \geq NH$ для любого целого $N > 0$. Положим $K = I = \{1, 2, \ldots, N\}$, $I_k = \{k\}$, $J = \{1\}$, $L = 1$, $c_k^\circ = c_k = c_{kj} = V_k = 0$, $p_{kj} = d_i = 1$, $v_k^\circ = 1/N$, $k \in K$, $j \in J$, $i \in I$. Тогда задача D_f может быть записана следующим образом: найти

$$\max_{\alpha}\Big\{\alpha + \min_{x,y}\Big\{\sum_{i\in I} y_i - \sum_{k\in K} \alpha\ x_k\Big\}\Big\}$$

при условиях:

$$x_k \leq v_k^\circ, \quad 0 \leq x_k \leq y_k, \quad y_k \in \{0,1\}, \quad k \in K.$$

Эта задача распадается по k на независимые подзадачи нахождения величины

$$g(\alpha) = \min_{x,y}\{y - \alpha x\}$$

при условиях:

$$Nx \leq 1, \quad 0 \leq\ x \leq y, \quad y \in \{0,1\}.$$

Следовательно, $F = \max\limits_{\alpha}\{\alpha + Ng(\alpha)\}$. Заметим, что $g(\alpha) = \min(0, 1 - \alpha/N)$, откуда $F = \max\limits_{\alpha}\{\alpha + \min(0, N - \alpha)\} = N$.

Вычислим значение H. На этих данных задача D_h может быть записана следующим образом: найти

$$H = \max_{\alpha,\beta}\Big\{\alpha + \min_{x,y}\Big\{\sum_{k\in K}(1-\beta_k)y_k + \sum_{k\in K}(\beta_k - \alpha)x_k\Big\}\Big\}$$

при условиях:

$$0 \leq Nx_k \leq 1, \quad y_k \in \{0,1\}, \quad k \in K.$$

Оптимальные значения переменных β_k равны 1 и $H = \max\limits_{\alpha}\{\alpha + \min(0; 1 - \alpha)\} = 1$.

2.3.3 Результаты тестовых расчетов

В предыдущем параграфе рассматривался пример исходных данных, на котором оптимальные значения всех переменных x_{kj} были достаточно близки к нулю. Такие примеры нетипичны. Дробные значения переменных x_{kj} появляются при жестких ограничениях (2.14) или малых значениях величин v_k°, $k \in K$. Ниже приводятся результаты тестовых расчетов, показывающие зависимость параметра $\delta = H/F$ от жесткости ограничений (2.13), (2.14).

Решаемые задачи имели размерность $m = 20$, $n = 20$, $q = 30$, $L = 4$. Значения величин выбирались следующим образом: $c_k^\circ = 50$, $c_k = 1$, $c_{kj} = s_{kl} = 0$, $d_i = 100$, $v_k^\circ = 0,25\lambda$, $V_k = \lambda$. Величины p_{kj} и множества I_k, $k \in K$, выбирались с помощью датчика псевдослучайных чисел с равномерным распределением. Множества I_k с вероятностью 0,2 содержали элементы множества I. Матрица (p_{kj}) на 30% заполнялась числом 10^6. Результаты расчетов изображены на рис. 4. Нумерация кривых соответствует следующим значениям величин p_{kj}:

$$\delta_1 \text{ при } p_{kj} \in [1, 10], \quad \delta_2 \text{ при } p_{kj} \in [2, 4], \quad \delta_3 \text{ при } p_{kj} = 3.$$

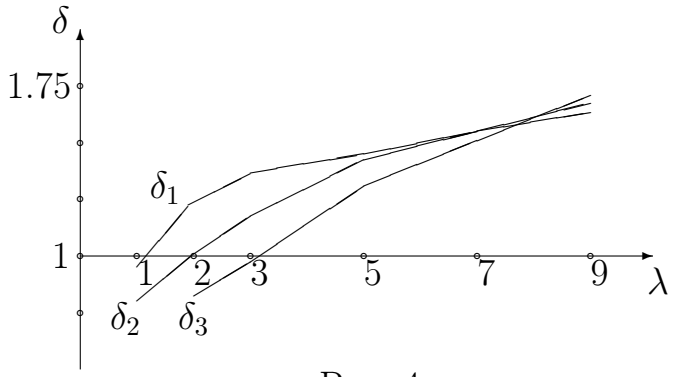


Рис. 4

Каждая точка излома на графике соответствует среднему геометрическому значению величины δ для 10 тестовых задач. Отметим, что при $\lambda < 1$ система ограничений исходной задачи несовместна. При $\lambda > 9$ ограничения (2.13), (2.14) становятся несущественными и исходная задача трансформируется в многостадийную задачу размещения. В целом можно утверждать, что нижняя оценка F имеет некоторые преимущества при жестких ограничениях (2.13), (2.14) и хуже нижней оценки H в остальных случаях.

2.4 Лагранжевы релаксации для задачи ВДСС

Предположим, что в течение определенного планового периода состав системы может меняться во времени за счет пополнения новыми образцами и изъятия части средств по причинам их морального или физического старения. Задача выбора динамического состава системы состоит в нахождении варианта развития системы, который доставлял бы минимум суммарных затрат на разработку, производство и эксплуатацию технических средств при условии выполнения заданного списка работ каждый год.

Используя введенные в разделе 1.5.3 обозначения, задача ВДСС может быть записана в следующей эквивалентной формулировке: найти

$$F_p = \min \sum_{i\in I} \sum_{t\in T} \left\{ c^0_{it} z_{it} + c_{it} v_{it} + \sum_{j\in J_t} c_{ij} x_{ij} \right\} \tag{2.20}$$

при условиях:

$$\sum_{i\in I} x_{ij} = 1, \quad j \in J, \tag{2.21}$$

$$\sum_{j\in J_t} p_{ij} x_{ij} \le v^0_{ij} y_{it} + \sum_{\tau=t-\rho_i+1}^{t} v_{i\tau}, \quad i \in I, \quad t \in T, \tag{2.22}$$

$$y_{it} \le \sum_{\tau\le t} z_{i\tau} \le 1,, \quad i \in I, \quad t \in T, \tag{2.23}$$

$$0 \le v_{it} \le V_{it} y_{it}, \quad i \in I, \quad t \in T, \tag{2.24}$$

$$0 \le x_{ij} \le y_{it(j)}, \quad i \in I, \quad j \in J, \tag{2.25}$$

$$y_{it}, z_{it} \in \{0, 1\}, \quad i \in I, \quad t \in T. \tag{2.26}$$

Для решения этой задачи приводятся алгоритмы, основанные на Лагранжевых релаксациях. Для решения релаксированных задач разработаны точные эффективные алгоритмы. Показано, что абсолютный разрыв двойственности для задачи ВДСС может быть сколь угодно большим. Выделяется подкласс задач, для которых двойственная задача D полиномиально разрешима. Рассматриваются обобщения исходной задачи на случай нелинейной зависимости стоимости производства изделий от размера партии и дополнительных ограничений на номенклатуру технических средств.

2.4.1 Нижние оценки

Рассмотрим Лагранжеву релаксацию $LR(\lambda)$ относительно условий (2.21): найти

$$\min \sum_{i\in I} \sum_{t\in T} \left\{ c^0_{it} z_{it} + c_{it} v_{it} + \sum_{j\in J_t} (c_{ij} - \lambda_j) x_{ij} \right\} + \sum_{j\in J} \lambda_j$$

при условиях: (2.22)–(2.26),

где $\lambda_j, j \in J$ — множители Лагранжа, соответствующие равенствам (2.21).

Теорема 9 *Оценочная задача $LR(\lambda)$ полиномиально разрешима.*

Доказательство. При фиксированных значениях $\lambda_j, j \in J$, задача $LR(\lambda)$ распадается по i на независимые подзадачи S_i следующего вида. Опуская индекс i, получаем: найти

$$\max \sum_{t \in T} \Big\{ \sum_{j \in J_t} (\lambda_j - c_j) x_j - c_t v_t - c_t^0 z_t \Big\}$$

при условиях:

$$\sum_{j \in J_t} p_j x_j \le v_t^0 y_t + \sum_{\tau = t-\rho+1}^{t} v_\tau, \quad t \in T,$$

$$y_t \le \sum_{\tau \le t} z_\tau \le 1, \quad t \in T,$$

$$0 \le v_t \le V_t y_t, \quad t \in T,$$

$$0 \le x_j \le y_{t(j)}, \quad j \in J,$$

$$y_t, z_t \in \{0, 1\}, \quad t \in T.$$

Для подзадач S_i справедливо равенство

$$v(LR(\lambda)) = \sum_{j \in J} \lambda_j - \sum_{i \in I} v(S_i).$$

Для каждой пары $(i, \theta), i \in I, \theta \in T$, рассмотрим задачу $S_{i\theta}$ линейного программирования следующего вида: найти

$$f_{i\theta} = \max \sum_{t \ge \theta} \Big\{ \sum_{j \in J_t} (\lambda_j - c_j) x_j - c_t v_t \Big\}$$

при условиях:

$$\sum_{j \in J_t} p_j x_j \le v_t^0 y_t + \sum_{\tau = t-\rho+1}^{t} v_\tau, \quad t \ge \theta,$$

$$0 \le v_t \le V_t, \quad t \ge \theta,$$

$$0 \le x_j \le 1, \quad j \in J_t, \quad t \ge \theta,$$

Из $z_t \in \{0, 1\}, t \in T$ и $\sum\limits_{t \in T} z_t \le 1$ следует, что

$$v(S_i) = \max\{0, \max_{\theta \in T} (f_{i\theta} - c_\theta^0)\},$$

откуда следует утверждение теоремы. $\square$

Теорема 10 *Если $\rho_i \geq l$, $i \in I$, то задача $LR(\lambda)$ разрешима с трудоемкостью $O(nm(l + \log n))$.*

Доказательство. Покажем, что задача S_i разрешима с трудоемкостью $O(n(l + \log n))$. Множества $J_t, t \in T$, упорядочим по невозрастанию величин $(\lambda_j - c_j)/p_j, j \in J_t$. Тогда значения переменных $x_j, j \in J$, однозначно восстанавливаются по значениям переменных v_t с линейной трудоемкостью. Оптимальные значения переменных $v_t, t \in T$, находятся с помощью процедуры координатного подъема. На каждом шаге этой процедуры ищется номер $\tau \in T$ такой, что увеличение целевой функции при возрастании переменной v_t достигает наибольшего значения. Затем определяется длина шага в направлении τ и изменяются значения соответствующих переменных. Если на некотором шаге ни по одной из координат нельзя добиться увеличения целевой функции, то полученное решение является оптимальным.

Упорядочение множеств $J_j, t \in T$, осуществляется с трудоемкостью $O(n \log n)$. Вычисление приращений и нахождение наилучшего номера $\tau \in T$ требует $O(l)$ операций. С такой же трудоемкостью находится и длина шага. Число шагов не превосходит величины $l+n$, и так как $l \leq n$, то общая трудоемкость алгоритма не превосходит величины $O(nl + n \log n)$.

□

Теорема 11 *Если $\rho_i = 1, i \in I$, то двойственная задача D полиномиально разрешима.*

Доказательство. Убедимся в том, что $v(\overline{S}_i) = v(S_i)$. Обозначим через (z_t^*), (y_t^*), (v_t^*), (x_j^*) оптимальное решение задачи $\overline{S}_i$. Для каждого $k \in T$ построим решение задачи S_i по следующему правилу:

$$z_t(k) = \begin{cases} 1, & \text{если } t = k, \\ 0 & \text{в противном случае,} \end{cases} \qquad t \in T,$$

$$y_t(k) = \begin{cases} 1, & \text{если } t \geq k, \\ 0 & \text{в противном случае,} \end{cases} \qquad t \in T,$$

$$v_t(k) = \begin{cases} 0, & \text{если } t < k, \\ v_t^*/y_t^* & \text{в противном случае,} \end{cases} \qquad t \in T,$$

$$x_j(k) = \begin{cases} 0, & \text{если } t < k, \\ x_j^*/y_{t(j)}^* & \text{в противном случае,} \end{cases} \qquad j \in J.$$

Если $z_k^* = 0$, то полагаем

$$z_t(k) = y_t(k) = v_t(k) = 0, \quad t \in T, \quad x_j(k) = 0, \quad j \in J.$$

Легко проверить, что для каждого $k \in T$ построенное решение является допустимым в задаче S_i. Кроме того,

$$v(\overline{S}_i) \le \sum_{t\in T} \Big\{ \sum_{j\in J_t} (\lambda_j - c_j) \sum_{k\in T} z_k^* x_j(k) - c_t \sum_{k\in T} z_k^* v_t(k) - c_t^0 \sum_{k\in T} z_k^* z_t(k) \Big\} =$$

$$\sum_{k\in T} z_k^* \sum_{t\in T} \Big\{ \sum_{j\in J_t} (\lambda_j - c_j) x_j(k) - c_t v_t(k) - c_t^0 z_t(k) \Big\} =$$

$$\sum_{k\in T} z_k^* v(S_i(k)) \le \max_k v(S_i(k)).$$

Таким образом, $v(\overline{S}_i) \le v(S_i)$. Справедливость обратного неравенства очевидна, откуда $v(\overline{S}_i) = v(S_i)$ и $v(LR(\lambda)) = v(\overline{LR}(\lambda))$ т. е. $v(D) = v(\overline{P})$. Следовательно, для решения задачи D достаточно найти оптимальное решение задачи линейного программирования $\overline{P}$ и по нему восстановить частично–целочисленное решение двойственной задачи. □

Отметим, что если условия теоремы не выполнены и $\rho > 1$ для некоторого $i \in I$, то абсолютный разрыв двойственности $v(\overline{S}_i) - v(S_i)$ может достигать значительных размеров.

Теорема 12 *Для любого $N > 0$ существуют исходные данные задачи S_i, на которых $v(\overline{S}_i) - v(S_i) > N$.*

Доказательство. Положим

$$T = \{1, 2\}, J = \{1, 2\}, J_1 = \{1\}, J_2 = \{2\},$$

$$p_j = 1, c_j = 0, j \in J, \quad V_t = 2N, v_t^0 = 0, t \in T,$$
$$\lambda = (0, N + 2), c^0 = (2N, 0), \ = (0, 2N), \rho = 2.$$

Легко проверить, что в задаче S_i нулевое решение является оптимальным, $v(S_i) = 0$, а на допустимом решении

$$z = (l/2N, l - l/2N); \quad y = (1/2N, 1); \quad v = (l, 0); \quad x = (0, 1)$$

значение целевой функции задачи $\overline{S}_i$ равно $N + 1$. □

2.4.2 Общая схема алгоритма

Алгоритм решения исходной задачи основан на полиномиальной разрешимости оценочной задачи $LR(\lambda)$. Работа алгоритма начинается с вычисления нижней оценки $v(D)$. Пусть $(z_{it}^k), (y_{it}^k), (v_{it}^k), (x_{ij}^k)$ — оптимальное решение задачи $LR(\lambda^k)$ при фиксированных значениях параметров $\lambda_j = \lambda_j^k, j \in J$. Если равенства (2.21) выполнены, то получено точное решение исходной задачи. В противном случае вектор

$$S_j^k = 1 - \sum_{i \in I} x_{ij}^k,\ j \in J,$$

является субградиентом функции $v(LR(\lambda))$ в точке $\lambda_j^k, j \in J$, и задает направление изменения множителей Лагранжа. Полагаем

$$\lambda_j^{k+1} = \lambda_j^k + \beta^k S_j^k, \quad j \in J,$$

и возвращаемся к решению задачи $LR(\lambda)$ с новыми значениями параметров λ [146].

Алгоритм построения верхней оценки F_0 формирует множество номеров $I_0 = \{i \in I \mid \sum\limits_k \sum\limits_{t \in T} z_{it}^k > 0\}$ и для каждого $i_0 \in I_0$ выполняет следующие действия.

Шаг 0. Положить $I_1 = \{i_0\}$.

Шаг 1. Решить двойственную задачу D с дополнительными ограничениями $\sum\limits_{t \in T} z_{it} = 1, i \in I_1$. На каждой итерации решения данной задачи оптимальное решение задачи $LR(\lambda)$ достроить до допустимого решения исходной задачи. Наилучшее решение запомнить в качестве рекорда F_0.

Шаг 2. Найти номер $i_1 \in I \setminus I_1$, для которого $\gamma = \sum\limits_k \sum\limits_{t \in T} z_{i_1 t}^k$ — частота вхождения в оптимальное решение — максимальна. Если $\gamma_{i_1} = 0$, то произвести переход к следующему элементу множества I_0. В противном случае положить $I_1 = I_1 \cup \{i_1\}$ и вернуться на шаг 1.

Трудоемкость алгоритма не превосходит величины $O(m^2 Q)$, где Q — трудоемкость решения двойственной задачи. О качестве получаемого решения можно судить, например, по оценке относительного уклонения $\varepsilon = (F_0/v(D) - 1)$, которая наряду с погрешностью самого алгоритма $\delta = (F_0/Opt - 1)$ включает еще и погрешность, связанную с разрывом двойственности.

2.4.3 Задачи с ограничениями на номенклатуру изделий

Предположим, что на развитие системы в течение планового периода наложены дополнительные ограничения, касающиеся номенклатуры технических средств. Пусть величины $b_{it}, i \in I, t \in T$, задают расход некоторого обобщенного ресурса, связанного с введением в состав системы новых образцов, а величина B — суммарный ресурс, выделяемый на развитие системы в течение всего планового периода. Дополнительная группа неравенств, соответствующая новым ресурсным ограничениям, может быть записана следующим образом:

$$\sum_{t \in T} \sum_{i \in I} b_{it} z_{it} \leq B. \tag{2.27}$$

В случае $b_{it} = 1, i \in I, t \in T$, неравенства (2.27) можно интерпретировать как ограничение на суммарное число образцов, применяемых для выполнения работ в течение всего планового периода. Задачу ВДСС с дополнительным ограничением (2.27) обозначим через P_B.

Рассмотрим другой вариант ограничений на номенклатуру технических средств, когда суммарный ресурс выделяется не сразу на весь интервал планирования, а разбивается по годам. Обозначим через $K_t, t \in T$, величину ресурса, выделяемого в t-м году. Тогда ограничения на номенклатуру образцов можно записать следующим образом:

$$\sum_{i \in I} b_{it} z_{it} \leq K_t, \quad t \in T. \tag{2.28}$$

Задачу с такими дополнительными ограничениями обозначим через P_K.

Построение алгоритмов решения задач P_B и P_K осуществляется по той же схеме, что и для исходной задачи. Решение задачи, двойственной к P_B, сводится в конечном счете к решению следующей задачи: найти

$$\max \sum_{\theta \in T} \sum_{i \in I} f_{i\theta} z_{i\theta}$$

при условиях:

$$\sum_{\theta \in T} z_{i\theta} \leq 1, \quad i \in I,$$

$$\sum_{\theta \in T} \sum_{i \in I} b_{i\theta} z_{i\theta} \leq B,$$

$$z_{i\theta} \in \{0, 1\}, \quad i \in I, \quad \theta \in T.$$

Эта задача известна как задача о многовариантном рюкзаке [109]. Линейная релаксация данной задачи решается с трудоемкостью $O(lm)$, и среди ее оптимальных решений всегда существует такое, в котором не более двух переменных принимают дробные значения. В случае $b_{it} = 1$, $i \in I, t \in T$, данная задача вырождается, и ее оптимальное решение находится за один просмотр.

Решение задачи, двойственной к P_K, сводится к решению следующей задачи: найти

$$\max \sum_{\theta \in T} \sum_{i \in I} f_{i\theta} z_{i\theta}$$

при условиях:

$$\sum_{\theta \in T} z_{i\theta} \leq 1, \quad i \in I,$$

$$\sum_{i \in I} b_{i\theta} z_{i\theta} \leq K_\theta, \quad \theta \in T,$$

$$z_{i\theta} \in \{0, 1\}, \quad i \in I, \quad \theta \in T.$$

Эта задача известна как обобщенная задача о назначениях [155]. В случае $b_{it} = 1, i \in I, t \in T$, она принимает вид транспортной задачи и легко решается точно. В общем случае она является NP-трудной, и для ее решения разработаны точные и приближенные алгоритмы [167, 211].

2.4.4 Задачи с фактором серийности

Рассмотрим обобщение исходной задачи, когда стоимость производства изделий $c_{it}, i \in I, t \in T$, не является постоянной величиной, а зависит от суммарного объема выпуска изделий

$$c_{it} = c_{it}\Big(\sum_{\tau \in T} v_{i\tau}\Big), \quad i \in I, t \in T.$$

Будем предполагать, что для каждого $i \in I$ заданы интервалы серийности w_i:

$$0 < w_{i1} < w_{i2} < \cdots < w_{iq_1} = \sum_{\tau \in T} V_{i\tau},$$

внутри каждого из которых стоимость производства остается постоянной величиной, а при переходе из одного интервала в другой может меняться произвольно.

Алгоритм решения такой задачи также основан на полиномиальной разрешимости релаксированной задачи $LR(\lambda)$, при построении которой задача $S_{i\theta}$ принимает следующий вид: найти

$$f_{i\theta} = \max \sum_{t \geq \theta} \Big\{ \sum_{j \in J_t} (\lambda_j - c_j) x_j - c_t \Big(\sum_{\tau \in T} v_{i\tau} \Big) v_t \Big\}$$

при условиях:

$$\sum_{j \in J_t} p_j x_j \leq v_t^0 + \sum_{\tau = t-\rho+1} v_\tau, \quad t \geq \theta,$$

$$0 \leq v_t \leq V_t, \quad t \geq \theta,$$

$$0 \leq x_j \leq 1, \quad j \in J_t,\, t \geq \theta.$$

Алгоритм решения данной задачи последовательно рассматривает все интервалы серийности $[w_{k-1}, w_k)$, $1 \leq l \leq q_i$, и на каждом из них решает следующую вспомогательную задачу: найти

$$f_k = \max \sum_{t \geq \theta} \Big\{ \sum_{j \in J_t} (\lambda_j - c_j) x_j - c_t(w_{k-1}, w_k) v_t \Big\}$$

при условиях:

$$\sum_{j \in J_t} p_j x_j \leq v_t^0 + \sum_{\tau = t-\rho+1} v_\tau, \quad t \geq \theta,$$

$$w_{k-1} \leq \sum_{t \geq \theta} v_t < w_k,$$

$$0 \leq v_t \leq V_t, \quad t \geq \theta,$$

$$0 \leq x_j \leq 1, \quad j \in J_t,\, t \geq \theta,$$

где $c_t(w_{k-1}, w_k)$ — постоянная величина, равная стоимости производства i-го образца в t-м году при объемах партии в интервале от w_{k-1} до w_k. Очевидно, что $f_{i\theta} = \max\limits_k f_k$.

При $\rho_i \geq 1$ эта задача упрощается и ее оптимальное решение находится с трудоемкостью $O(nq_i(l + n \log n))$. В общем случае она является задачей линейного программирования и решается стандартными методами.

Предположим, что стоимость производства изделий на каждом интервале серийности задается линейной функцией $c_t(v) = c_t^0(w_{k-1}, w_k) + c_t(w_{k-1}, w_k)v_t, t \in T$. В этом случае последняя задача не меняет своего вида, но величина $f_{i\theta}$ находится с помощью равенства

$$f_{i\theta} = \max_k \Big(f_k + \sum_{t \geq \theta} c_t^0(w_{k-1}, w_k) \Big).$$

2.4.5 Результаты тестовых расчетов

Разработанные алгоритмы реализованы на языке ПАСКАЛЬ и тестировались при следующей размерности: $m = 50$, $n = 50$, $l = 5$, $n_t = 10$, $t \in T$.

Массивы исходных данных формировались с помощью датчика псевдослучайных чисел с равномерным распределением. Значения величин выбирались следующим образом: $c^0_{it} = 300c^0$, $c_{it} \in [1, 10]$, $c_{ij} \in [1, 1000]$, $p_{ij} \in [1, 10]$, $b_{it} = 1$, $v^0_{it} = 0$. Матрицы (c_{ij}) и (p_{ij}) на 30% заполнялись числом 10^6. Результаты расчетов изображены на рис. 5. Погрешности δ_1 и ε_1 вычислялись при $\rho_i = 1$, $V_{it} = 100$, $B = 3$, величины δ_2 и ε_2 — при $\rho_i = 1$, $V_{it} = 100$, $B = 50$.

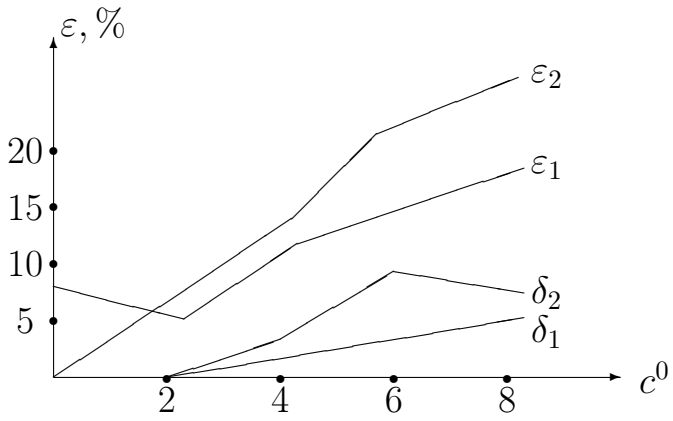


Рис. 5

Каждая точка излома является средним значением относительной погрешности для 30 тестовых задач. Отметим, что при $c^0 = 0$ исходная задача остается обобщением NP-трудной задачи о K-медиане [163]. В этом случае при $B = 3$ разрыв двойственности составляет заметную величину (от 7 до 15%), а предлагаемый алгоритм почти не ошибается ($\delta = 0, 16\%$).

При больших значениях параметра c^0 исходная задача вырождается и фактически становится задачей о минимальном покрытии. Начальные затраты доминируют в целевой функции, и если алгоритму удается найти минимальную номенклатуру для выполнения всех работ, то получается точное решение задачи. В противном случае отклонение от оптимума достигает 10%.

Время решения одной задачи при $\rho_i = 1$ в среднем составляет 2–3 минуты, при $\rho_i = 5$ — от 10 до 15 мин. При промежуточных значениях ρ_i время решения, по-видимому, будет больше. В связи с этим возникает

потребность в специализированных алгоритмах для задачи $S_{i\theta}$ при произвольных $\rho_i, i \in I$. Кроме того, на практике часто возникает ситуация, когда часть технических средств ежегодно выходит из строя еще до окончания срока эксплуатации. В этом случае ограничения (2.22) принимают вид

$$\sum_{j \in J_t} p_{ij} x_{ij} \leq v_{ij}^0 y_{it} + \sum_{\tau=t-\rho_i+1}^{t} s_{i\tau} v_{i\tau}, \quad i \in I,\ t \in T,$$

где величины $s_{i\tau}$ задают долю изъятия технических средств в течение срока эксплуатации.

Другим важным обобщением являются динамические задачи с суммарными ограничениями на ежегодные объемы производства изделий или (и) на суммарный состав системы в каждом году. Для таких моделей требуется как минимум существенная переработка алгоритмов или разработка принципиально других подходов.

Глава 3

Метаэвристики

3.1 Метаэвристики для задач комбинаторной оптимизации

Развитие численных методов решения NP–трудных задач идет в целом по двум направлениям: точные и приближенные методы. Мы не будем касаться точных методов, хотя это очень интересная и быстро развивающаяся область (см., например [194]). Остановимся подробнее на приближенных методах. Фактически здесь имеются две больших ветви: алгоритмы с гарантированными оценками точности и эвристики. Первое направление касается, как правило, полиномиальных алгоритмов, для которых удается установить оценку погрешности в худшем случае или в среднем. Этому посвящены сотни статей и монографий с блестящими математическими результатами (см., например [80, 160]). Однако на практике такие подходы редко находят применение. Для многих задач комбинаторной оптимизации и, в частности, дискретных задач размещения, погрешность любого полиномиального алгоритма не может быть ограничена константой. Это обстоятельство подталкивает к разработке неполиномиальных алгоритмов, среди которых наибольшей эффективностью отличаются метаэвристики. Сам термин метаэвристика был предложен в [128] для обозначения достаточно общих эвристических схем решения задач комбинаторной оптимизации и не только их. В отличие от алгоритмов с оценками, метаэвристики не привязываются к специфике решаемой задачи. Это достаточно общие итерационные процедуры, использующие рандомизацию и элементы самообучения, интенсификацию и диверсификацию поиска, адаптивные механизмы управления, конструктивные эвристики и методы локального поиска. Нет и, наверное, не может быть точного определения метаэвристики. Попытки дать такое

определение можно найти в [93]. К метаэвристикам принято относить методы имитации отжига *Simulated Annealing* (SA), поиск с запретами *Tabu Search* (TS), генетические алгоритмы *Genetic Algorithms* (GA) и эволюционные методы *Evolutionary Computation* (EC), а также поиск с чередующимися окрестностями *Variable Neighborhood Search* (VNS), муравьиные колонии *Ant Colony Optimization* (ACO), вероятностные жадные алгоритмы *Greedy Randomized Adaptive Search Procedure* (GRASP) и др. [105, 71, 98, 119]. Идея этих методов основана на предположении, что целевая функция решаемой задачи имеет много локальных экстремумов, а просмотр всех допустимых решений невозможен, несмотря на конечность их числа. В такой ситуации нужно сосредоточить поиск в наиболее перспективных частях допустимой области. Таким образом, задача сводится к выявлению таких областей и быстрому их просмотру. Каждая из метаэвристик решает эту проблему по-своему.

Метаэвристики принято делить на траекторные методы, когда на каждой итерации имеется одно допустимое решение и осуществляется переход к следующему, и на методы, работающие с семейством (популяцией) решений. К первой группе относятся TS, SA, VNS, GRASP. Траекторные методы оставляют в пространстве поиска траекторию, последовательность решений, где каждое решение является соседним для предыдущего относительно некоторой окрестности. Понятие окрестности играет здесь центральную роль. В методах TS, SA окрестность определяется заранее и не меняется в ходе работы. Целевая функция вдоль траектории меняется немонотонно, что позволяет "выбираться" из локальных экстремумов и находить всё лучшие и лучшие приближенные решения задачи. Элементы самоадаптации позволяют менять управляющие параметры алгоритмов, используя предысторию поиска. Более сложные методы, например, VNS, используют несколько окрестностей и меняют их систематически в целях диверсификации. Фактически при смене окрестности происходит смена ландшафта [200, 201]. Сознательное изменение ландшафта, как, например, это происходит в методе шума (Noising method, [99]), благотворно сказывается на результатах поиска. Изучение ландшафтов, их свойств, например, изрезанности (ruggedness) позволяет давать рекомендации по выбору окрестности для траекторных методов ([78], см. также [200, 201]).

Ко второй группе методов относятся GA, EC, ACO и др. [105, 71, 98]. На каждой итерации этих методов строится новое решение задачи, которое основывается уже не на одном, а на нескольких решениях из по-

пуляции. В генетических и эволюционных алгоритмах для этих целей используются процедуры скрещивания (crossover) и целенаправленных мутаций. В методах ACO применяется другая идея, основанная на сборе статистической информации о наиболее удачных найденных решениях. Эта информация учитывается в вероятностных жадных алгоритмах и подсказывет, какие именно компоненты решений (ребра графа, предприятия, элементы системы технических средств) чаще всего приводили ранее к малой погрешности. Методы этой группы, как правило, основаны на аналогиях в живой природе. Идея ACO является попыткой имитации поведения муравьев, которые почти не имеют зрения и ориентируются по запаху, оставленному предшественниками. Сильно пахнущее вещество, феромон, является для них индикатором деятельности предшественников. Оно аккумулирует в себе предысторию поиска и подсказывает дорогу к муравейнику. Попытки подглядеть у природы способы решения трудных комбинаторных задач находят все новые и новые воплощения в численных методах. Например, при инфекции организм старается подобрать (породить, сконструировать) наиболее эффективную защиту. Наблюдение за этим процессом привело к рождению новой метаэвристики — искусственных иммунных систем [98]. Исследование жизнедеятельности пчелиного улья, где только одна матка оставляет потомство, послужило основой для новых генетических методов [116]. Однако наибольший прогресс в области метаэвристик наблюдается на пути их гибридизации, построения гиперэвристик [98], автоматически подбирающих наиболее эффективные эвристики для данного примера и симбиоз с классическими методами математического программирования.

Остановимся подробнее на последнем направлении. Здесь наблюдается достаточно широкий спектр исследований:

– построение экспоненциальных по мощности окрестностей, в которых поиск наилучшего решения осуществляется за полиномиальное время путем решения вспомогательной оптимизационной задачи [73, 136, 137];

– исследование операторов скрещивания, базирующихся на точном или приближенном решении исходной задачи на подмножестве, заданном родительскими решениями [72, 85, 86, 27];

– гибридизация метаэвристик с точными методами, например, с методами ветвей и границ [21, 129] и динамического программирования [208].

– разработка новых точных методов, использующих идеи локального поиска [121];

– применение разных математических формулировок решаемой за-

дачи и использование различных кодировок решений в метаэвристиках [172, 159] и др. Обзор достижений в данном направлении можно найти в [93].

В нижеследующих разделах этой главы будут показаны возможности применения метаэвристик при решении дискретных задач размещения. Вероятностные жадные алгоритмы исследуются на примере многостадийной задачи размещения. Два варианта таких алгоритмов : "Лидер группы" и "Случайный аутсайдер", последний из которых основан на приближенном решении линейной релаксации задачи, экспериментально исследуются на четырех классах тестовых примеров. Установлен характер зависимости относительной погрешности получаемых решений от параметров алгоритмов. Показано, что вероятность получения точного решения задачи может быть сделана сколь угодно близкой к единице. Определены статистические значения математического ожидания погрешности, её среднеквадратичного отклонения, а также вероятность нахождения такого решения и приближенного решения с погрешностью не более одного процента. Показано, что вероятностные алгоритмы имеют лучшие характеристики, чем их детерминированные аналоги, хотя и являются более трудоемкими.

Новый вариант алгоритма поиска с запретами исследуется на примере дискретных задач безусловной оптимизации. Как показано в первой главе, многие дискретные задачи размещения сводятся к этому классу. Найдены условия на параметры предлагаемого алгоритма, при которых вероятность получения точного решения задачи стремится к единице с ростом числа итераций. Показано влияние параметров алгоритма на качество получаемых решений.

Алгоритмы генетического локального поиска исследуются применительно к задаче о p-медиане с предпочтениями клиентов. В качестве популяции в этом подходе используется некоторое множество локальных оптимумов относительно новой мощной окрестности типа Лина–Кернигана. Для оценки качества получаемых решений используются сведения исходной задачи к задаче целочисленного линейного программирования. Предложено новое сведение, доминирующее уже известные по значению целевой функции линейной релаксации.

Возможности построения гибридных алгоритмов иллюстрируются на конкурентной задаче о p-медиане. Предложен гибридный алгоритм, сочетающий в себе достоинства генетических алгоритмов и траекторных методов, в частности, вероятностного поиска с запретами. Показаны воз-

можности применения данного подхода к точному решению задачи.

3.2 Вероятностные жадные алгоритмы

В данном разделе для решения многостадийной задачи размещения представлены новые вероятностные жадные алгоритмы «лидер группы» и «случайный аутсайдер», приведены их характеристики и указаны возможные пути модификации этих алгоритмов.

Напомним, что многостадийная задача размещения может быть сформулирована в виде задачи поиска минимума следующей целевой функции:

$$\sum_{i\in I} f_i \max_{p\in P_i}\{y_p\} + \sum_{p\in P} f_p^0 y_p + \sum_{j\in J} \min_{p\in P}\{c_{pj} \mid y_p = 1\}.$$

Пусть $S = \{p \in P \mid y_p = 1\}$. Тогда эту целевую функцию можно представить следующим образом

$$F(S) = \sum_{i:S\cap P_i\neq\emptyset} f_i + \sum_{p\in S} f_p^0 + \sum_{j\in J} \min_{p\in S} c_{pj},$$

и задача сводится к поиску непустого подмножества $S \subseteq P$, при котором функция $F(S)$ принимает наименьшее значение.

Одним из наиболее простых и естественных способов приближенного решения этой задачи является алгоритм координатного спуска (КС). Он начинает свою работу с нулевого решения. На каждой итерации алгоритма выбирается некоторый элемент $p \in P$ и соответствующая переменная y_p полагается равной единице. Выбор элемента осуществляется по принципу наибольшего уменьшения значения целевой функции. Если на некоторой итерации добавление любого элемента не приводит к уменьшению целевой функции, то алгоритм заканчивает свою работу. Обозначим через $\rho_l = F(S) - F(S \cup \{l\})$ изменение целевой функции при добавлении l-го элемента и положим $M = \sum\limits_{p\in P} f_p^0 + \sum\limits_{i\in I} f_i + \sum\limits_{j\in J} \max\limits_{p\in P} c_{pj}$. Алгоритм координатного спуска заключается в следующем.

Алгоритм КС

1. Положить $S = \emptyset,\ F(\emptyset) = M$.
2. Вычислить $\rho_p,\ p \in P$.
3. Найти элемент l с наибольшим приращением $\rho_l = \max\{\rho_p \mid p \in P\}$.

Если $\rho_l \leq 0$, то перейти на шаг 5.

4. Добавить элемент l в множество S и вернуться на шаг 2.
5. Положить $F = F(S)$.

Алгоритм KC является детерминированным и полиномиальным. Его относительная погрешность

$$\varepsilon = (F - F^*)/F^*, \quad \text{где } F^* \text{ — оптимум задачи,}$$

в худшем случае растет не медленнее логарифмической функции от размерности задачи, если верно предположение о несовпадении классов NP и $DTIME(m^{O(\log\log m)})$ [117]. Однако в среднем относительная погрешность алгоритма невелика.

Для исследования этого и других алгоритмов решения задачи выделим четыре класса ($R,\ R_0,\ E,\ E_0$) тестовых примеров:

$$\begin{array}{rlll} R: & f_p^0 = 3000, & f_i = 3000, & c_{pj} \in [0, 1000]; \\ R_0: & f_p^0 = 0, & f_i = 3000, & c_{pj} \in [0, 1000]; \\ E: & f_p^0 = 3000, & f_i = 3000, & c_{pj} \text{— евклидовы расстояния;} \\ E_0: & f_p^0 = 0, & f_i = 3000, & c_{pj} \text{— евклидовы расстояния.} \end{array}$$

Все примеры имеют следующую размерность: $n = 100,\ m = 50,\ |P| = 100,\ |I_p| = 4,\ p \in P$. Различие между классами состоит в матрице (c_{pj}) и векторе (f_p^0). В классах R, R_0 величины c_{pj} выбираются случайно с равномерным распределением на интервале [0,1000]. В классах E, E_0 величины c_{pj} являются евклидовыми расстояниями на плоскости между точками p и j. Координаты точек выбираются случайно с равномерным распределением в квадрате 1000×1000.

Указанные классы задач являются трудными для метода ветвей и границ [19] и требуют для точного решения от 10 до 120 мин машинного времени (здесь и далее время счета указано для PC Pentium-166). Если величины f_p^0, f_i выбирать случайно из некоторого интервала, то тестовые примеры становятся проще. Выбор константы 3000 принципиального значения не имеет. Увеличение константы приводит к сокращению множества открываемых предприятий и технологических цепочек, уменьшение константы — к его расширению.

В каждом классе взято 30 примеров. Для каждого из них найдено точное решение методом ветвей и границ. Все приближенные алгоритмы тестировались на этих примерах, и результаты сравнивались с точными решениями. В итоге для каждого алгоритма получены статистические

оценки следующих характеристик:

$M(\varepsilon)$ — математического ожидания относительной погрешности ε,

$D(\varepsilon)$ — ее среднего квадратического отклонения (квадратный корень из дисперсии),

$Pr(\varepsilon = 0)$ — вероятности получения точного решения,

$Pr(\varepsilon \leq 0,01)$ — вероятности получения приближенного решения с относительной погрешностью не более одного процента, а также среднего времени работы алгоритмов.

Для классов $R,\ R_0,\ E,\ E_0$ статистические характеристики алгоритма KC (в %) приведены в табл. 1. Среднее время работы алгоритма — 0,02 с.

Таблица 1. *Характеристики алгоритма KC*

Характеристика	R	R_0	E	E_0
$M(\varepsilon)$	3,38	7,34	2,06	10,22
$D(\varepsilon)$	2,77	4,77	2,91	10,87
$Pr(\varepsilon = 0)$	16,6	10,0	30,0	0
$Pr(\varepsilon \leq 0,01)$	26,6	10,0	46,6	0

3.2.1 Алгоритм «лидер группы»

Приведем описание вероятностной версии алгоритма КС, которая имеет в среднем меньшую погрешность и большую вероятность получения точного решения. Единственное отличие этого алгоритма от предыдущего состоит в том, что на шаге 2 величины ρ_p вычисляются не для всего множества P, а только для его части P', выбранной случайно из множества $P \setminus S$. Пусть величина q задает вероятность включения номера $p \in P$ в подмножество P'. Вероятностный алгоритм, назовем его «лидер группы» (ЛГ), может быть записан следующим образом.

Алгоритм ЛГ

1. Положить $S = \emptyset,\ F(\emptyset) = M$.

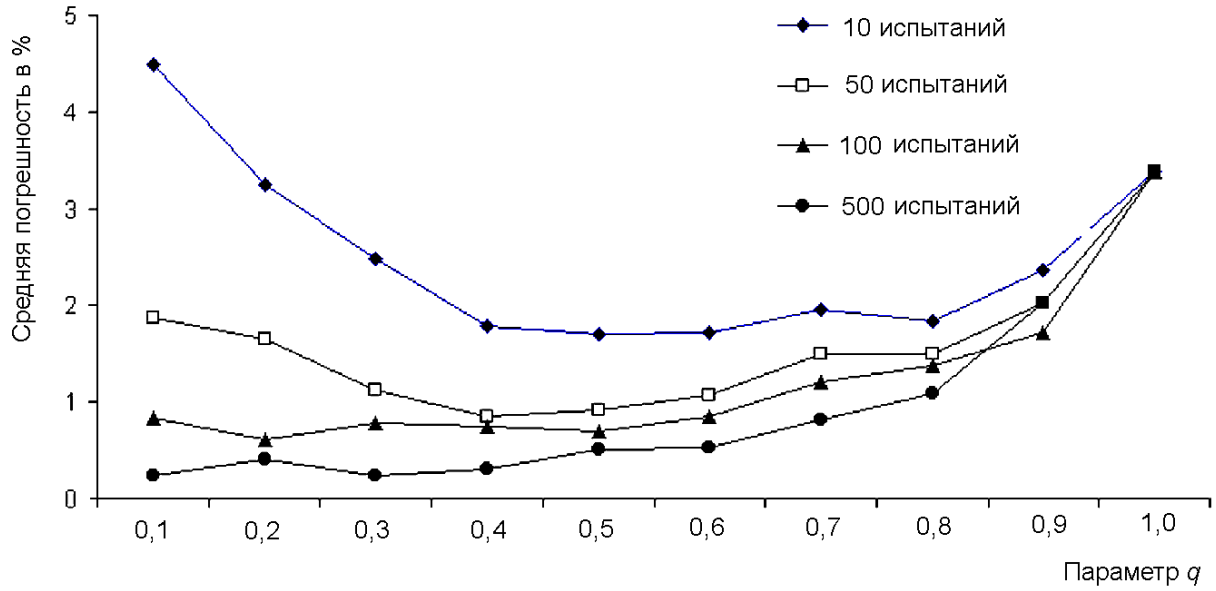
2. Сформировать случайным образом подмножество P' и вычислить ρ_p,

Если $\rho_l \leq 0$ или $P' = \emptyset$, то перейти на шаг 5.

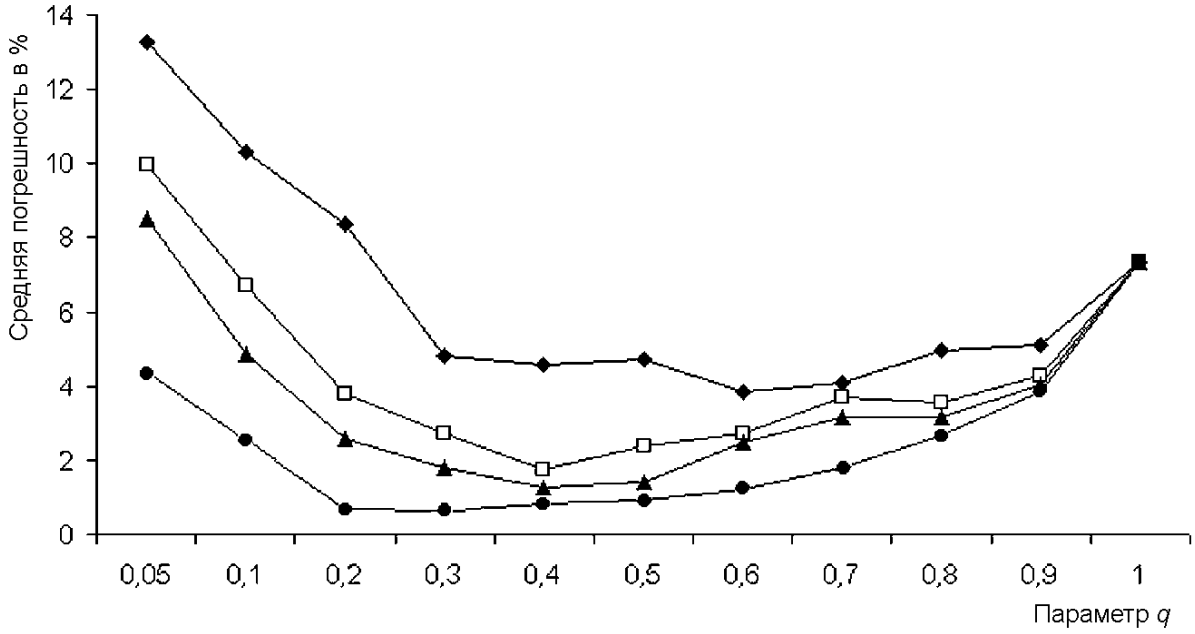
4. Добавить элемент l в множество S и вернуться на шаг 2.
5. Положить $F = F(S)$.

Результатом работы алгоритма является случайная величина $F(q)$. Применяя алгоритм k раз и выбирая из найденных решений наилучшее, получаем величину $F(q,k)$. Очевидно, что $F(1,k) = F$. Если $q < 1$ и число испытаний k достаточно велико, то погрешность такого вероятностного алгоритма оказывается меньше погрешности детерминированного алгоритма (рис. 6).

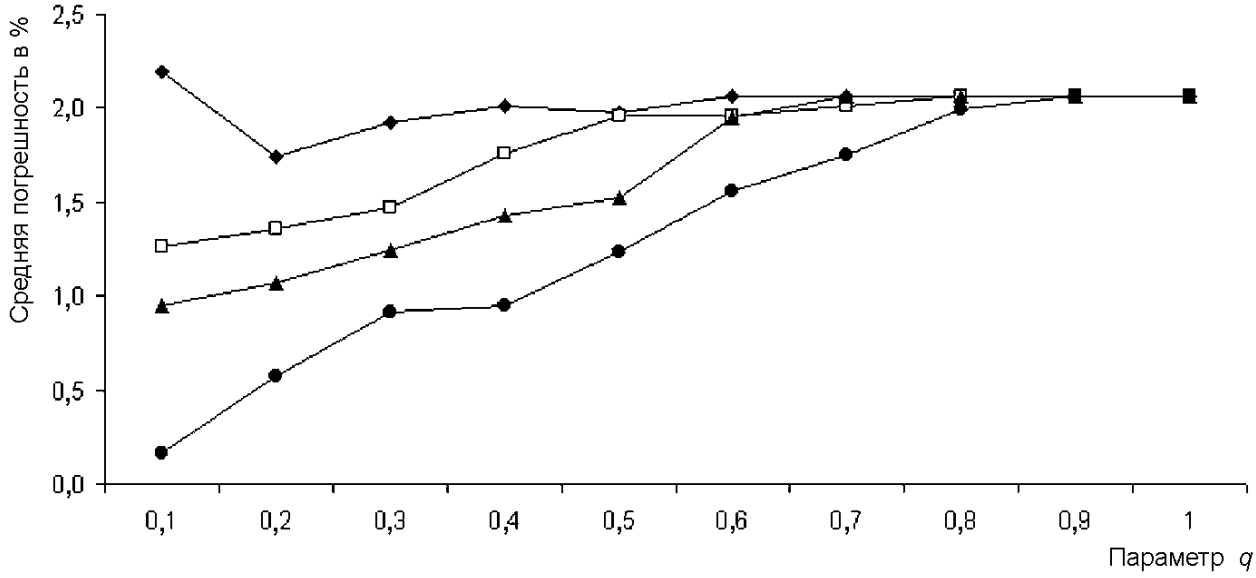
При $k = 500$ и $0,2 \leq q \leq 0,7$ величина $F(q,k)$ существенно меньше значения F. Интересно отметить, что характер зависимости средней относительной погрешности алгоритма от вероятности q различен для классов R, R_0 и E, E_0. График этой зависимости напоминает выпуклую функцию для классов R, R_0. Для классов E, E_0 при $q \geq 0,2$ зависимость от q монотонная.



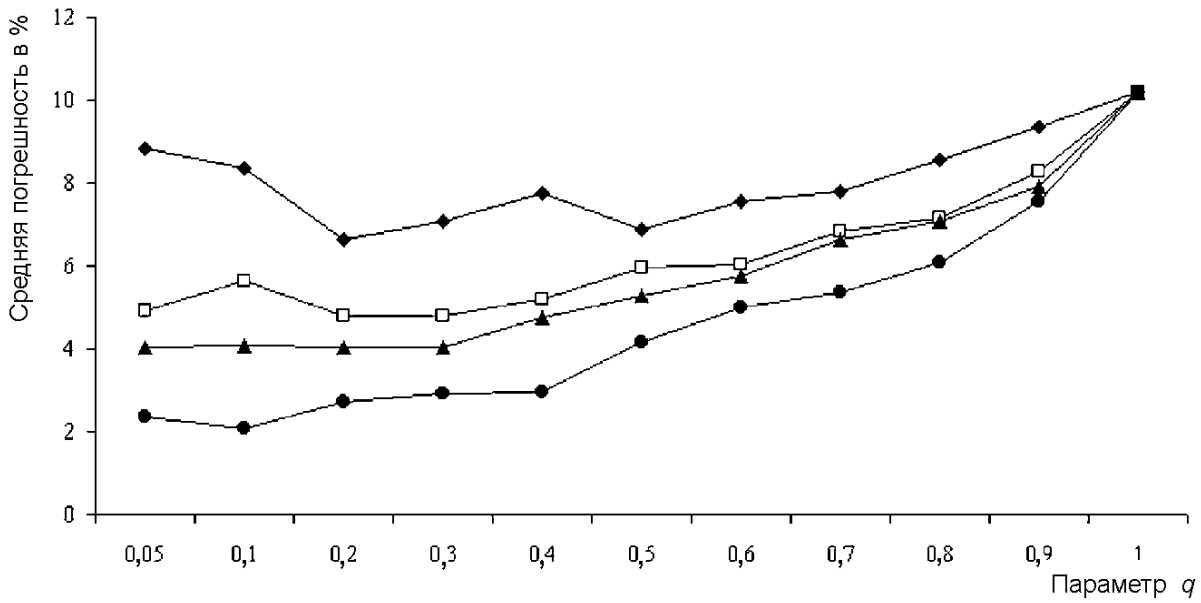
а) класс R



б) класс R_0



в) класс E



г) класс E_0

Рис. 6. *Зависимость средней погрешности алгоритма ЛГ от параметра* q

В среднем погрешность алгоритма на классе E_0 всегда больше, чем на классе E, и погрешность на классе R_0 больше, чем на классе R. В этом смысле классы E_0 и R_0 более трудные для жадных алгоритмов, чем классы E и R.

На каждой итерации алгоритма ЛГ случайным образом формируется множество P' и для каждого элемента этого множества вычисляется приращение целевой функции. В среднем этот шаг алгоритма в $1/q$ раз менее трудоемкий, чем шаг алгоритма КС. Многократные испытания алгоритма ЛГ увеличивают время его работы, но позволяют сократить погрешность и увеличить вероятность нахождения точного решения. При $k = 100$ характеристики алгоритма для классов задач R, R_0, E, E_0 значительно превосходят соответствующие показатели детерминированного алгоритма КС (табл. 2). Среднее время работы алгоритма 1,5 с.

Таблица 2. *Характеристики алгоритма ЛГ* ($k = 100$)

Характеристика	R	R_0	E	E_0
$M(\varepsilon)$	0,61	1,27	0,95	4,03
$D(\varepsilon)$	1,15	1,67	2,06	5,23
$Pr(\varepsilon = 0)$	66,6	49,0	53,3	13,6
$Pr(\varepsilon \leq 0,01)$	80,0	62,3	80,0	23,9

Критерии завершения работы алгоритмов ЛГ и КС не являются самыми «жадными» из возможных. Если на некотором шаге алгоритма $\rho_p \le 0,\ p \in P$, то отсюда не следует, что множество S нельзя расширить с улучшением значения целевой функции. Рассмотрим следующий пример. Пусть $|J| = |P| = 3,\ \ |I| = 4,\ \ f_p^0 = 0,\ \ f_i = 3/5,$ $P_i = \{p \in P \mid q_{pi} = 1\}$ и

$$(q_{pi}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad (c_{pj}) = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Алгоритм КС сначала выберет $l = 1$ и получит $S = \{1\},\ \ F(S) = 0 + 0,6+0+1+1 = 2,6$. На следующем шаге добавление любого из элементов $l = 2,\ l = 3$ приводит к увеличению целевой функции, так как $\rho_2 = \rho_3 = -0,2$. Включение же сразу двух элементов дает оптимальное решение: $F^* = 2,4;\ \ S = \{1, 2, 3\}$.

Критерий окончания работы алгоритмов можно изменить. Например, в качестве нового критерия можно взять условие $P' = \varnothing$, а результатом работы считать наилучшее из найденных решений. Такая модификация делает алгоритм ЛГ асимптотически точным, т. е. вероятность получения оптимального решения стремится к единице с ростом числа испытаний [21]. Однако на практике это свойство часто оказывается бесполезным, поскольку может потребоваться слишком много испытаний даже при малой размерности задачи. Тем не менее, относительная погрешность действительно заметно падает уже при небольшом числе испытаний. Алгоритм порождает спектр разных приближенных решений задачи, которые могут и действительно оказываются полезными для методов локального поиска. В схеме GRASP к каждому такому решению применяются процедуры локального улучшения [118]. В итоге получается семейство локальных оптимумов, которое служит основой для построения начальной популяции в эволюционных алгоритмах [164, 174]. В [185, 186] такой подход позволил с малой (часто нулевой) погрешностью решать задачи размещения большой размерности.

3.2.2 Условия дополняющей нежесткости

Заменим условия булевости переменных на условия их неотрицательности и получим следующую задачу линейного программирования: найти

$$\min\Big\{\sum_{i\in I} f_i x_i + \sum_{p\in P} f_p^0 y_p + \sum_{j\in J}\sum_{p\in P} c_{pj}x_{pj}\Big\}$$

при условиях:

$$\sum_{p\in P} x_{pj} = 1, \quad j \in J,$$

$$\sum_{p\in P_i} x_{pj} \le x_i, \quad i \in I,\ j \in J,$$

$$x_{pj} \le y_p, \quad p \in P,\ j \in J,$$

$$x_{pj},\ x_i,\ y_p \in \{0,1\}, \quad i \in I,\ p \in P,\ j \in J.$$

Тогда двойственная задача имеет вид: найти

$$\max_{u,v}\sum_{j\in J}\min_{p\in P}\left(c_{pj} + u_{pj} + \sum_{i\in I_p} v_{ij}\right)$$

при условиях:

$$\sum_{j\in J} v_{ij} \le f_i, \quad i \in I,$$

$$\sum_{j\in J} u_{pj} \le f_p^0, \quad p \in P,$$

$$u_{pj} \ge 0, \quad v_{ij} \ge 0, \quad p \in P, \quad i \in I, \quad j \in J.$$

Условия дополняющей нежесткости для этих задач записываются следующим образом:

$$x_i\left(\sum_{j\in J} v_{ij} - f_i\right) = 0, \quad i \in I, \quad y_p\left(\sum_{j\in J} u_{lj} - f_p^0\right) = 0, \quad p \in P.$$

Из этих условий следует, что если $(v_{ij},\ u_{lj})$ — оптимальное решение двойственной задачи и для некоторых i, p неравенства строгие, то в оптимальном решении прямой задачи должно быть $y_p = x_i = 0$. Эта информация о решении непрерывной задачи может оказаться полезной при решении дискретной задачи.

В работах [6, 10, 19] исследовалось поведение приближенных алгоритмов решения двойственной задачи. Эти алгоритмы имеют полиномиальную трудоемкость и строят так называемые «тупиковые» решения,

имеющие небольшую погрешность (около $10-15\%$ на классах задач $R,\ R_0,\ E,\ E_0$). Тупиковое решение $(v_{ij},\ u_{lj})$ обладает тем свойством, что его нельзя улучшить за счет только увеличения значений переменных $v_{ij},\ u_{lj}$. Точное определение тупикового решения дано в работах [9, 10].

Пусть $(v_{ij},\ u_{pj})$ — тупиковое решение. Обозначим через Δf_i и Δf_p^0 невязки в ограничениях двойственной задачи:

$$\Delta f_i = f_i - \sum_{j\in J} v_{ij}, \quad i \in I, \quad \Delta f_p^0 = f_p^0 - \sum_{j\in J} u_{pj}, \quad p \in P.$$

Положим

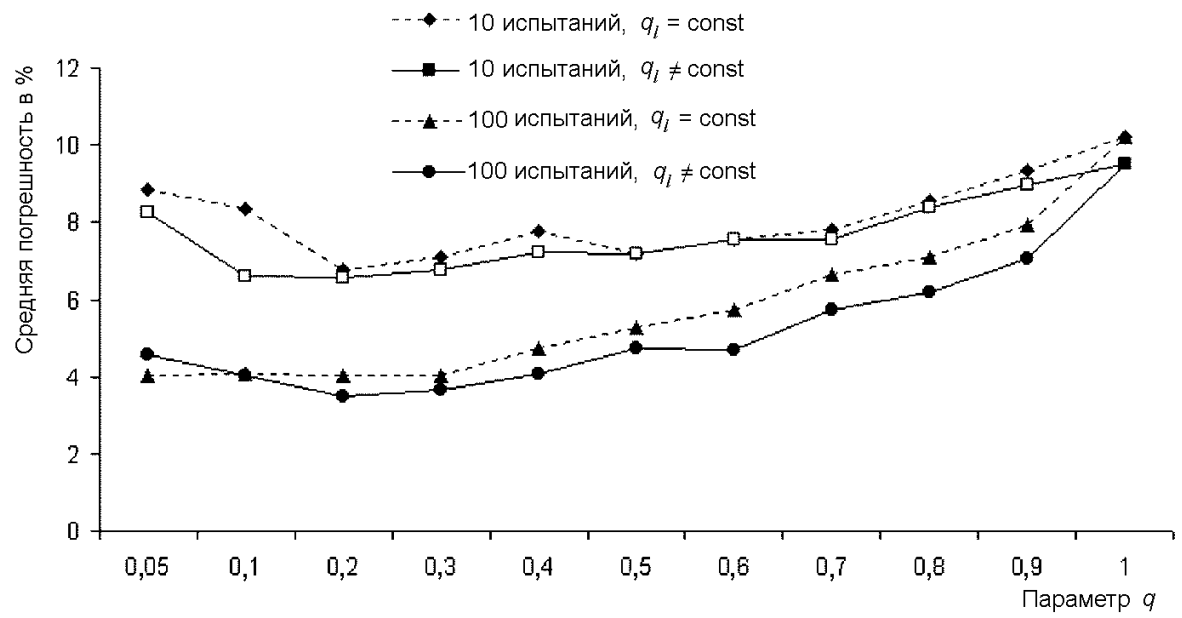
$$q'_p = q\left(1 - \left(\Delta f_p^0 + \sum_{i\in I_p} \Delta f_i\right) \Big/ \left(f_p^0 + \sum_{i\in I_p} f_i\right)\right)^3, \quad p \in P.$$

Рассмотрим модификацию алгоритма ЛГ, в которой вместо константы q используется вектор $(q'_p),\ p \in P$. Смысл такой замены состоит в том, чтобы уменьшить вероятность рассмотрения переменных y_p с ненулевой невязкой в ограничениях.

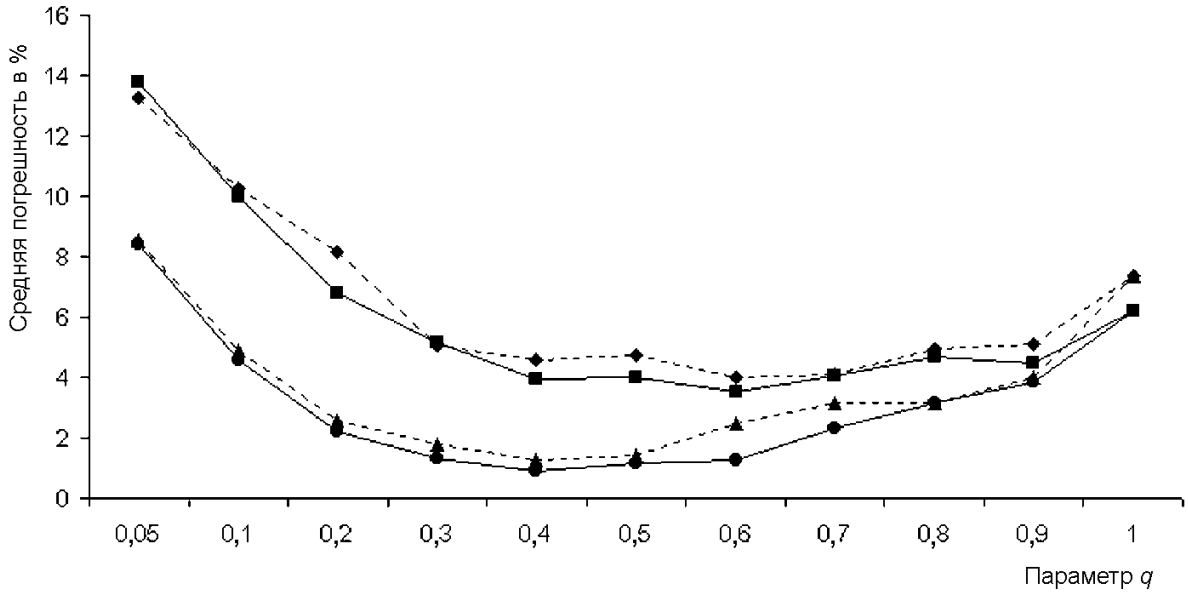
Влияние этой замены на погрешность алгоритма изображено на рис. 7. Средняя погрешность действительно уменьшается, но говорить о кардинальных изменениях нельзя. Столь незначительное уменьшение погрешности (в среднем около 0,5 %) связано в первую очередь с большим разрывом двойственности $\delta = 100\%(F^* - \overline{F})/F^*$, где $\overline{F}$ — оптимум релаксированной задачи. На классах $R,\ R_0,\ E,\ E_0$ эта величина составляла

$$\delta_R \approx 15,2\%, \quad \delta_{R_0} \approx 33,55\%, \quad \delta_E \approx 7,06\%, \quad \delta_{E_0} \approx 24,14\%,$$

что, по-видимому, ослабляло влияние условий дополняющей нежесткости на работу алгоритма.



а) класс E_0



б) класс R_0

Рис. 7. *Влияние невязок на погрешность алгоритма ЛГ*

3.2.3 Алгоритм «случайный аутсайдер»

С каждой технологической цепочкой $p \in P$ свяжем дополнительное предприятие, стоимость открытия которого равна f_p^0. Этот прием позво-

ляет свести исходную задачу к случаю, когда стоимость открытия любой цепочки равна нулю. Далее считаем такое сведение выполненным.

Вектору (x_i) поставим в соответствие множество $K = \{i \in I \mid x_i = 1\}$ и положим $S = \{p \in P \mid I_p \subseteq K\}$. Минимум целевой функции по переменным x_{pj} при фиксированном векторе (x_i) равен величине

$$F(K) = \sum_{i \in K} f_i + \sum_{j \in J} \min_{p \in S} c_{pj},$$

если $S \neq \emptyset$. Полагаем $F(K) = M$, если $S = \emptyset$. Таким образом, многостадийную задачу размещения можно представить как задачу минимизации функции $F(K)$ по непустым подмножествам $K \subseteq I$.

Предлагаемый ниже алгоритм, используя тупиковое решение двойственной задачи и соответствующие невязки Δf_i, строит начальное решение $K_1 = \{i \in I \mid \Delta f_i = 0\}$ и затем пытается его улучшить с помощью вероятностной жадной процедуры.

Обозначим через $\rho_i = F(K) - F(K \setminus \{i\})$ приращения функции $F(K)$ на текущем решении K. Введем в рассмотрение параметры $\varphi,\ \psi \in [0, 1]$, и определим множества

$$I'(\varphi) = \{i \in I \mid 0 < \Delta f_i < \varphi f_i\}, \qquad I''(\psi) = \Big\{i \in K \mid \rho_i \geq \psi \max_{k \in K} \rho_k\Big\}.$$

Алгоритм «случайный аутсайдер» (СА) последовательно сокращает множество K, случайным образом удаляя из него элементы множества $I''(\psi)$ (список аутсайдеров), и через заданное число шагов $\xi \geq 2$ расширяет множество K за счет элементов множества $I'(\varphi)$ (список претендентов).

Алгоритм СА

1. Положить $K = K_1,\ K_2 = I'(\varphi),\ t = 0$.
2. Вычислить $\rho_i,\ i \in K$. Если $\max\limits_{i \in K} \rho_i \leq 0$, то перейти на шаг 6.
3. Сформировать множество $I''(\psi)$, выбрать случайным образом элемент $i'' \in I''(\psi)$ и удалить его из множества K.
4. Если $t < \xi$, то положить $t = t + 1$ и вернуться на шаг 2.
5. Случайным образом выбрать элемент $i' \in K_2$ и перенести его из множества K_2 в множество K. Установить $t = 0$ и вернуться на шаг 2.
6. Положить $F = F(K)$.

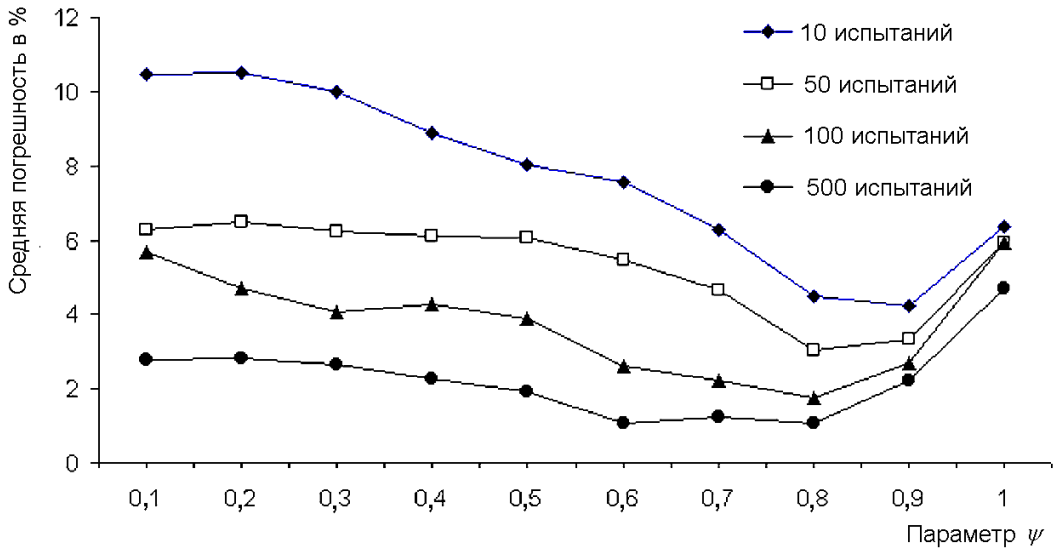
Результатом работы алгоритма «случайный аутсайдер» является случайная величина F. За счет ее дисперсии можно уменьшать погрешность

приближенного решения задачи, увеличивая число испытаний и выбирая из найденных решений наилучшее.

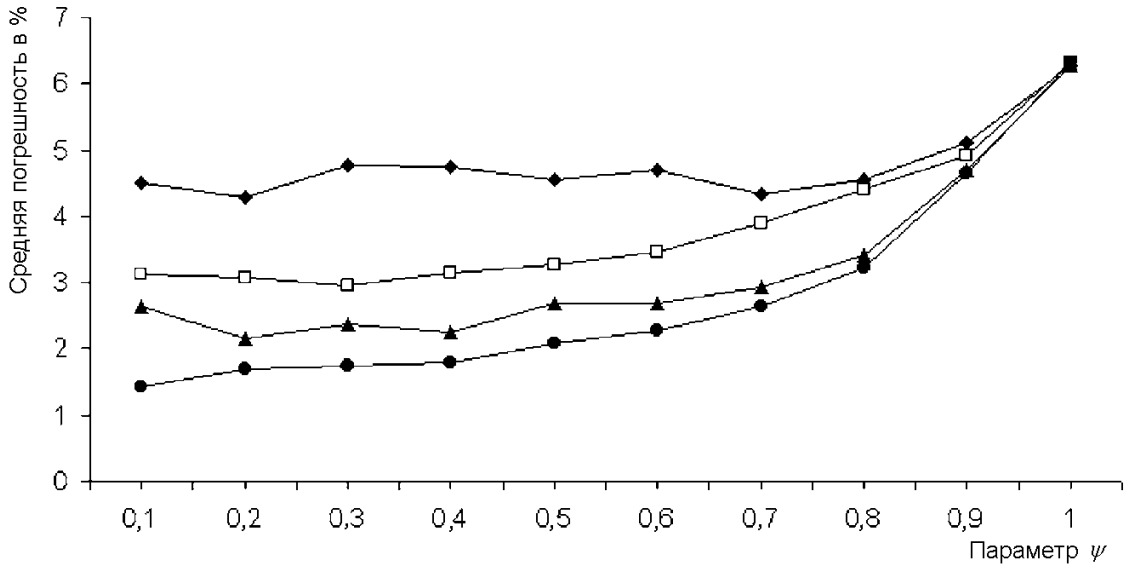
Алгоритм СА устроен сложнее алгоритма ЛГ. Во-первых, сокращая множество K, алгоритм через каждые ξ шагов расширяет это множество. Так как $\xi \geq 2$, то это мало влияет на трудоемкость алгоритма, но позволяет повысить его точность. Во-вторых, этот алгоритм использует известное правило выбора направления спуска, предложенное в [17]. Согласно этому правилу на шаге 3 алгоритм формирует множество $I''(\psi)$, которое может быть пустым только в том случае, когда все приращения ρ_i неположительны. В этом смысле он действует аккуратнее алгоритма ЛГ.

Алгоритм СА не является асимптотически точным, но его можно сделать таковым, изменив правило выбора множества K на первом шаге. Пусть величины p_1 и p_2 задают вероятности включения в множество K элементов множеств K_1 и $I \setminus K_1$. Тогда с ненулевой вероятностью уже на первом шаге алгоритма может быть получено оптимальное решение задачи, и он становится асимптотически точным.

Средняя погрешность алгоритма при различных значениях параметра ψ изображена на рис. 8. Значения $\varphi = 0,3$ и $\xi = 4$ выбраны близкими к наилучшим. Характеристики алгоритма (в процентах) для $k = 100$, $p_1 = 1$, $p_2 = 0$ и лучших значений параметра ψ приведены в табл. 3. Среднее время работы алгоритма равно 6,7 с.



а) класс R_0



б) класс E_0

Рис. 8. *Влияние невязок на погрешность алгоритма СА*

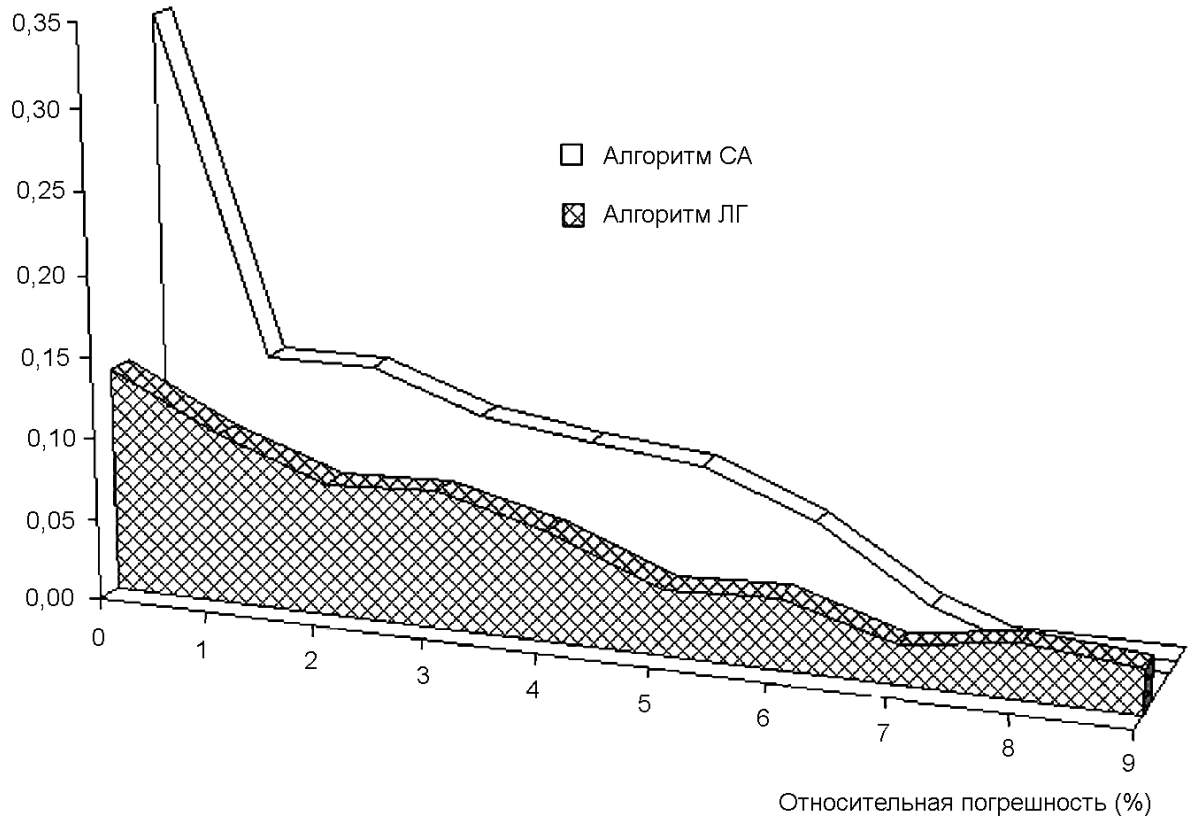
Таблица 3. *Характеристики алгоритма СА* ($\varphi = 0,3,\ \xi = 4,\ k = 100$)

Характеристика	R	R_0	E	E_0
$M(\varepsilon)$	2,02	1,77	2,21	2,16
$D(\varepsilon)$	1,67	1,75	2,42	2,34
$Pr(\varepsilon = 0)$	23,3	45,6	13,3	34,0
$Pr(\varepsilon \leq 0,01)$	36,6	57,2	50,0	47,3

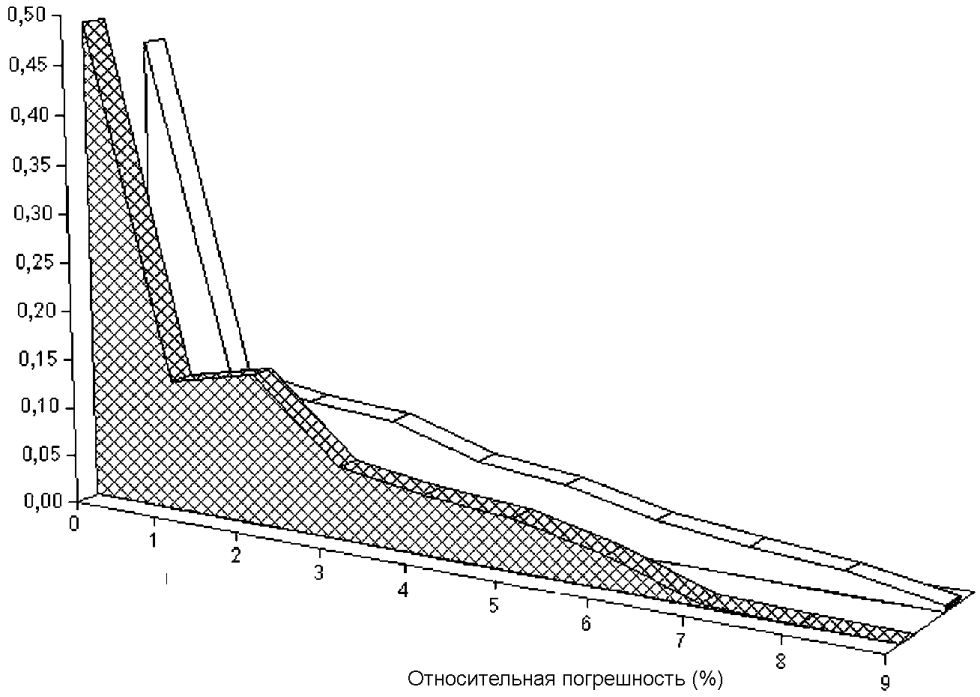
Алгоритм СА в среднем ведет себя не хуже алгоритма ЛГ. На классе E_0 он имеет явное преимущество по всем показателям, но на классах R, E уступает алгоритму ЛГ.

На рис. 9 изображены графики распределения вероятностей относительной погрешности алгоритмов на классах задач E_0 и R_0. Значения в нуле соответствуют доле числа задач, решенных точно. Значения в точке i соответствуют доле числа задач, для которых относительная погрешность оказалась в интервале от $i-1$ до i процентов. Для повышения достоверности результатов каждая из тридцати тестовых задач решалась

вероятностными алгоритмами 10 раз. Из рис. 9 видно, что распределения вероятностей не подчиняются нормальному закону. Это обстоятельство затрудняет получение доверительных интервалов для относительной погрешности алгоритмов. Тем не менее, каков бы ни был закон распределения, всегда имеется возможность вычислить доверительные интервалы для вероятностей $Pr(\varepsilon = 0)$ и $Pr(\varepsilon_1 < \varepsilon < \varepsilon_2)$.



а) класс E_0



б) класс R_0

Рис. 9. *Распределения вероятностей для относительной погрешности алгоритмов СА и ЛГ*

Рассмотрим случайную величину $x \in \{0, 1\}$, принимающую значение 1, если алгоритм получил точное решение задачи, и 0 в противном случае. Обозначим через p вероятность того, что $x = 1$, и положим $q = 1 - p$. Пусть $x_1, \ldots, x_N$ — результаты независимых испытаний алгоритма. Интервал (γ_1, γ_2) называется доверительным интервалом с коэффициентом доверия $1 - \delta$, если

$$Pr(\gamma_1 < p < \gamma_2) = 1 - \delta.$$

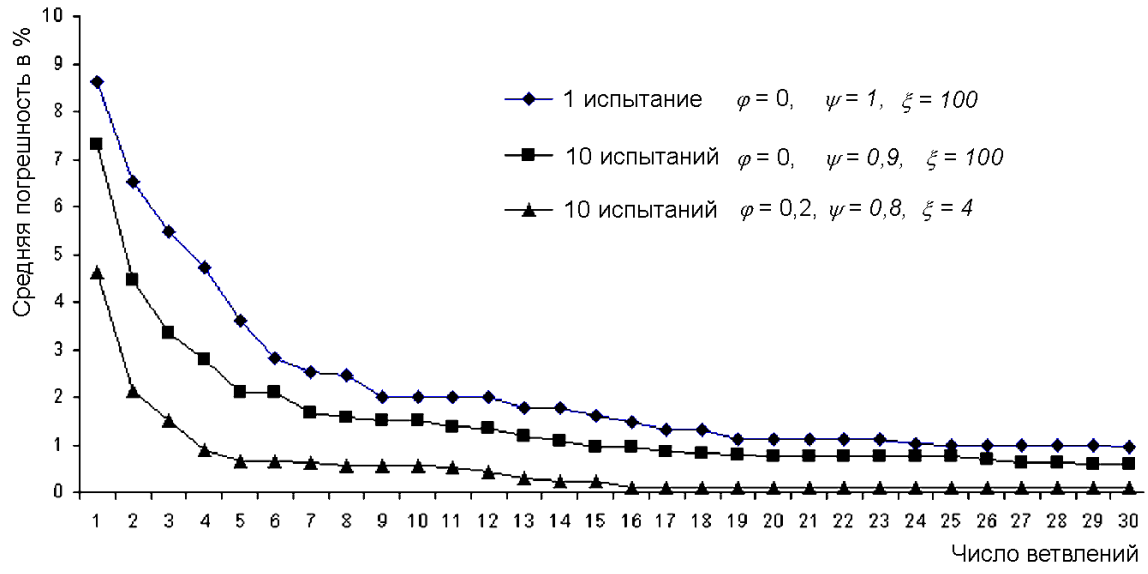
Согласно [44] величина $p^* = \sum\limits_i x_i/N$ является эффективной оценкой, которая асимптотически нормальна с параметрами $(p, \sqrt{pq/N})$. Полагая $\gamma_1 = p^* - \lambda(\delta, N)\sqrt{p^*(1-p^*)/N}$ и $\gamma_2 = p^* + \lambda(\delta, N)\sqrt{p^*(1-p^*)/N}$, получаем доверительный интервал для p. При $\delta = 0,05$, $N \geq 120$ величина $\lambda(\delta, N)$ полагается равной 1,96. Аналогично строятся доверительные интервалы для вероятности $Pr(\varepsilon_1 < \varepsilon < \varepsilon_2)$.

3.2.4 Принцип ветвления

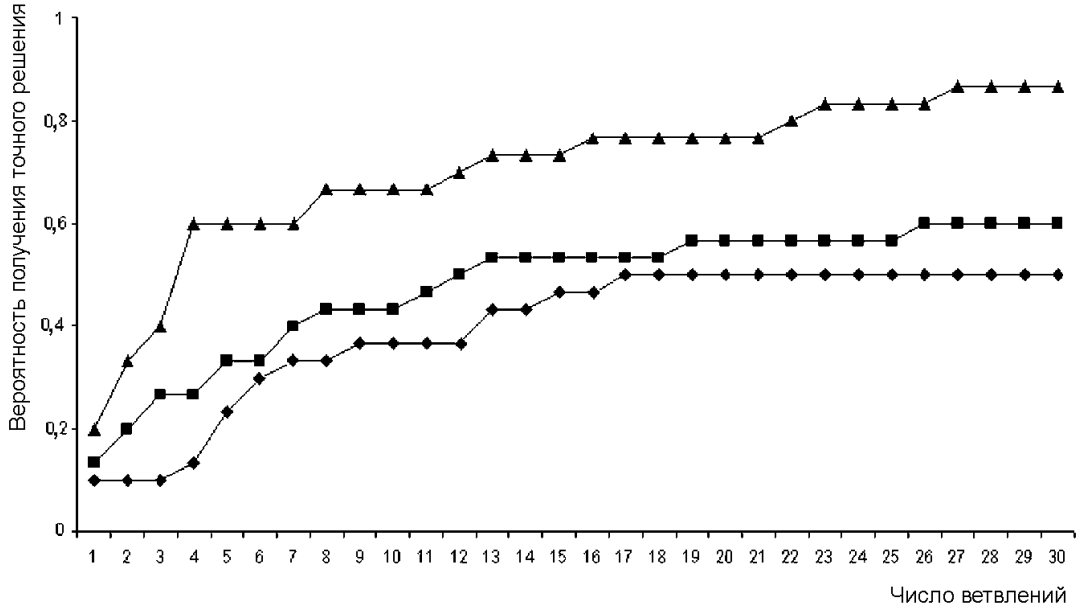
В работе [19] исследовалось поведение метода ветвей и границ (МВГ) для многостадийной задачи размещения. Этот метод позволяет нахо-

дить точное решение за приемлемое время при размерностях $I \leq 50$, $L \leq 100$, $J \leq 100$. Для алгоритма характерно быстрое нахождение точного решения и затем длительная проверка того, что лучшего решения не существует. Однако с ростом размерности задачи картина меняется. Алгоритм быстро находит хорошее приближенное решение задачи, но предсказать, когда будет найдено точное решение, уже затруднительно. Общая схема МВГ ориентирована на достижение одновременно двух целей: найти оптимальное решение и проверить его оптимальность. Вторая цель является трудно достижимой. Что же касается первой цели, то кратчайший путь к ней может лежать в сочетании идей ветвления и вероятностных эвристик.

Рассмотрим приближенный полиномиальный алгоритм, который выполняет несколько первых итераций МВГ и на каждой итерации использует вероятностные жадные алгоритмы. Впервые такой подход был предложен в [21]. Ключевую роль здесь играют схема ветвления, выбор переменной для ветвления и применяемый вероятностный жадный алгоритм.



а) изменение средней относительной погрешности $M(\varepsilon)$



б) изменение частоты получения точного решения $Pr(\varepsilon = 0)$

Рис. 10. *Изменение характеристик алгоритма СА на классе* R_0 *с ростом числа ветвлений*

Пусть $(x_i,\ x_{pj})$ — допустимое решение, полученное на текущей итерации МВГ. Как и прежде, $K = \{i \in I \mid x_i = 1\}$ и $\rho_i = F(K) - F(K \setminus \{i\})$, $i \in K$. В качестве переменной для ветвления будем брать такую переменную x_t, для которой $\rho_t = \min\limits_{i \in K} \rho_i$, и использовать это правило в многосторонней схеме ветвления.

Изменение характеристик алгоритма СА с ростом числа итераций МВГ изображено на рис. 10. Верхняя кривая (рис. 10*а*) соответствует погрешности детерминированного алгоритма координатного спуска, нижняя — погрешности алгоритма СА при наилучшем выборе параметров, средняя кривая — погрешности алгоритма СА в случае чистого координатного спуска ($I'(\varphi) = \varnothing$). На первых итерациях МВГ погрешность всех алгоритмов резко падает и уже к 10-й итерации сокращается более чем в три раза. После 20-й итерации эффективность ветвления уменьшается и погрешность меняется медленно. На классе R_0 первых 10 ветвлений оказалось достаточно, чтобы значительно сократить погрешность и резко увеличить вероятность получения точного решения (рис. 10*б*). С ростом числа ветвлений растет и число испытаний алгоритма. Интересно отметить, что если число испытаний зафиксировать, например, положить $k = 600$, то средняя погрешность на классе R_0 составит без

ветвления 1,16%, а с ветвлением — 0,09%, т. е. станет на порядок меньше.

Таблица 4. *Характеристики алгоритмов* ЛГ и СА *с ветвлением*

Характеристика	Алгоритм ЛГ				Алгоритм СА			
	R	R_0	E	E_0	R	R_0	E	E_0
$M(\varepsilon)$	0,40	0,16	0,53	1,20	0,27	0,36	0,89	0,92
$D(\varepsilon)$	0,64	0,28	1,36	1,18	1,02	0,13	1,27	1,41
$Pr(\varepsilon = 0)$	77,0	88,0	67,7	55,3	78,0	75,7	43,7	58,7
$Pr(\varepsilon \leq 0,01)$	86,6	90,7	86,3	67,3	88,8	84,3	70,3	73,4

В табл. 4 приведены характеристики вероятностных алгоритмов при 10 ветвлениях (21 вершина в дереве ветвления). В каждой вершине вероятностные алгоритмы подвергались 10 испытаниям. Сравнивая эти данные, можно заметить, что разница в результатах незначительная.

Таблица 5. *Доверительные интервалы для* $Pr(\varepsilon = 0)$

Алгоритм	R	R_0	E	E_0
ЛГ	(72,2; 81,8)	(84,3; 91,7)	(62,4; 73,0)	(49,7; 60,9)
СА	(73,1; 82,7)	(70,9; 80,5)	(38,1; 49,3)	(53,1; 63,3)

Процедура ветвления позволила улучшить все показатели алгоритмов и сгладить их различия. Среднее время работы алгоритмов ЛГ и СА равно 8,5 и 25 с. В табл. 5 приведены доверительные интервалы для вероятности получения точного решения задачи при коэффициенте доверия 0,95. Таким образом, сочетание вероятностных эвристик с ветвлением приводит к лучшим результатам, чем «простое» увеличение числа испытаний.

3.2.5 Модификации алгоритмов ЛГ и СА

Вероятностные жадные алгоритмы являются одним из наиболее быстрых и эффективных способов приближенного решения NP-трудных задач дискретной оптимизации. Эти алгоритмы просто устроены, легко адаптируются к изменениям модели. Мы подробно рассмотрели алгоритмы «лидер группы» и «случайный аутсайдер». Эти алгоритмы по-

разному используют элемент рандомизации в своей работе. Первый алгоритм случайным образом формирует подмножество претендентов и в нем находит лучший элемент «на добавление». Второй алгоритм просматривает все элементы и среди них случайно выбирает один «на удаление». Первый алгоритм работает с множеством технологических цепочек, второй — с множеством предприятий. Кроме того, второй алгоритм случайным образом формирует начальное множество и время от времени случайно «расширяет» это множество. Применяя данные приемы рандомизации, легко построить другие вероятностные алгоритмы. Например, работа первого алгоритма могла бы начинаться с частичного решения, построенного с помощью условий дополняющей нежесткости. Элемент «на добавление» мог бы выбираться случайно из всего множества координат. Работа второго алгоритма могла бы опираться не на множество предприятий, а на множество цепочек, и т. д. Комбинируя эти приемы, можно построить семейство вероятностных жадных алгоритмов и использовать их параллельно для повышения точности решения задачи.

Вероятностные жадные алгоритмы имеют хорошие показатели качества вследствие многократности испытаний. При этом испытания алгоритмов проходили независимо друг от друга. Одним из способов повышения точности решения задачи является организация совместной работы алгоритмов на основе зависимых испытаний, использующих анализ предшествующей «коллективной» работы [1]. Другое перспективное направление исследований — сочетание вероятностных жадных алгоритмов с итеративными асимптотически точными метаэвристиками: имитацией отжига, поиском с запретами, генетическими алгоритмами [71] и др. Возможно, что именно эти направления позволят быстро и с высокой вероятностью находить оптимальные решения NP-трудных задач.

3.3 Вероятностный поиск с запретами

Алгоритм поиска с запретами относится к классу алгоритмов локального поиска и является одним из наиболее эффективных средств для решения задач дискретной оптимизации [129]. Он был предложен Ф. Гловером и хорошо показал себя при решении задач размещения [22, 158, 90, 171], теории расписаний [159, 77], календарного планирования [43] и др. [105, 98, 129]. Его несомненными достоинствами являются легкая адаптация к сложным моделям, простота и возможность гибридизации с другими методами, например, методом ветвей и границ, Лагранжевых

релаксаций и т. д. В настоящем разделе предлагается новый вариант алгоритма поиска с запретами, который порождает однородную непериодическую цепь Маркова с конечным множеством состояний. Установлены условия на параметры алгоритма, при которых эта цепь является неразложимой. Данные свойства гарантируют сходимость по вероятности наилучшего найденного алгоритмом решения к глобальному оптимуму. Для исследования поведения алгоритма проведены численные эксперименты на трудных в вычислительном отношении примерах многостадийной задачи размещения. Показано влияние на качество получаемых решений таких параметров алгоритма, как степень рандомизации окрестности, длина списка запретов и стратегии выбора новых областей для повышения эффективности поиска.

3.3.1 Общая схема

Рассмотрим задачу минимизации целевой функции $F(x)$ на гиперкубе $E^n = \{0,1\}^n$: найти

$$\min\{F(x) \mid x \in E^n\}.$$

Обозначим через $d(x,y)$ расстояние Хемминга между x и y. Через $N_k(x)$ обозначим окрестность точки x радиуса k, т. е.

$$N_k(x) = \{y \in E^n \mid d(x,y) \le k\}, \quad k = 1,2,\ldots,n.$$

При $k = n$ множество $N_k(x)$ любой точки x совпадает с E^n и нахождение в этой окрестности точки с минимальным значением целевой функции равносильно решению исходной задачи. Поэтому далее будут рассматриваться только малые значения $k = 1,2$ как наиболее удобные для локального поиска, хотя все последующие рассуждения можно обобщить и на случай произвольного $k < n$. Стандартный алгоритм локального спуска начинает свою работу с произвольно выбранной точки x^0. На t-м шаге алгоритма осуществляется переход из текущей точки в соседнюю точку, которая имеет минимальное значение целевой функции в окрестности данной точки, т. е.

$$F(x^{t+1}) = \min\{F(y) \mid y \in N_k(x^t)\}.$$

Алгоритм заканчивает работу в точке локального оптимума, когда $F(x^{t+1}) = F(x^t)$. При решении задач дискретной оптимизации типична ситуация, когда число локальных оптимумов велико и только один

из них является глобальным оптимумом

$$F_{opt} = \min\{F(y) \mid y \in E^n\}.$$

Для того чтобы алгоритм не останавливался в точке локального оптимума, а *путешествовал* от одного локального оптимума к другому, из окрестности удаляется центральная точка и при поиске минимума применяется следующее правило. Пусть $k = 2$ и при переходе от x^{t-1} к x^t меняются значения в координатах (i_t, j_t). Алгоритм хранит такие пары за последние l шагов и на следующем шаге запрещает движение в данных направлениях. Упорядоченный список таких пар

$$\varphi^t = \{(i_t, j_t), (i_{t-1}, j_{t-1}), \ldots, (i_{t-l+1}, j_{t-l+1})\}$$

называется списком запретов. По построению в списке запретов все пары разные, и пара (i, j), $i \neq j$, не запрещает движение по парам $(i, i), (j, j)$ и наоборот. Для удобства будем считать, что $i_t \leq j_t$. Если на t-м шаге менялось значение только одной координаты, то $i_t = j_t$. При $k > 2$ аналогичным образом строятся тройки координат, четверки и т. д. Список запретов удаляет из окрестности $N_k(x^t)$ ровно l точек. Множество незапрещенных точек обозначим через $N_k(x^t, \varphi^t)$. Для того чтобы поиск был эффективным, целесообразно использовать малые значения l и управлять данным параметром в ходе работы алгоритма. Ниже будут приведены границы для l, при которых вероятностный алгоритм поиска с запретами находит глобальный оптимум за достаточно большое число шагов.

Обозначим через $N_k(x^t, \varphi^t, p)$ вероятностную окрестность, которая получается из детерминированной окрестности $N_k(x^t, \varphi^t)$ следующим способом. Каждая точка $y \in N_k(x^t, \varphi^t)$ с вероятностью p включается в окрестность $N_k(x^t, \varphi^t, p)$ независимо от других точек. Заметим, что с ненулевой вероятностью это множество может оказаться пустым или содержать только одну точку. Теперь общая схема алгоритма поиска с запретами может быть представлена следующим образом.

Вероятностный алгоритм поиска с запретами

1. Выбрать $x^0 \in E^n$ и положить $F_A = F(x^0)$, $\varphi^0 = \emptyset$, $t = 0$.
2. Выполнять, пока не сработал критерий остановки.

2.1. Сформировать окрестность $N_k(x^t, \varphi^t, p)$.

2.2. Если $N_k(x^t, \varphi^t, p) = \emptyset$, то положить $x^{t+1} = x^t$, иначе найти такую точку x^{t+1}, что $F(x^{t+1}) = \min\{F(y) \mid y \in N_k(x^t, \varphi^t, p)\}$.

2.3. Если $F(x^{t+1}) < F_A$, то изменить значение $F_A := F(x^{t+1})$.

2.4. Обновить список запретов φ^t и счетчик $t := t + 1$.

3. Предъявить наилучшее найденное решение.

В качестве критерия остановки может использоваться требуемая точность по отношению к заданной нижней или верхней оценке глобального оптимума, остановка по общему числу шагов либо числу шагов, в ходе которых не меняется значение F_A. Величины p и l являются управляющими параметрами алгоритма. Их выбор зависит от специфики оптимизационной задачи и ее размерности. Варианты этой схемы и их адаптацию к различным NP-трудным задачам можно найти, например, в [129, 148].

3.3.2 Асимптотические свойства

Пусть $l = 0$ и $p < 1$. В этом случае за счет рандомизации окрестности алгоритм имеет шанс *покинуть* локальный оптимум даже без списка запретов. Выбор точки x^{t+1} зависит только от текущей точки x^t и не зависит от предыстории, т. е. последовательность $\{x^t\}$ является цепью Маркова с конечным множеством состояний E^n. Для любых двух точек $x, y \in E^n$ существует положительная вероятность перехода из x в y за конечное число шагов. Следовательно, цепь Маркова является неразложимой и

$$\Pr\{F_A = F_{opt}\} \to 1 \text{ при } t \to \infty.$$

Пусть $l > 0$. Обозначим через Ω множество всех пар (x, φ) таких, что $x \in E^n, \varphi$ — список запретов. Это множество конечно, и переход от одной пары к другой также не зависит от предыстории, т. е. последовательность $\{(x^t, \varphi^t)\}$ также является конечной цепью Маркова.

Теорема 13 *При любых l и p таких, что $0 < l < n(n-1)/4$ и $0 < p < 1$, вероятностный алгоритм поиска с запретами порождает неразложимую цепь Маркова.*

Доказательство. Пусть $(x, \xi) \in \Omega$ и $(y, \phi) \in \Omega$. Построим путь от (x, ξ) до (y, ϕ) такой, чтобы на каждом шаге вдоль этого пути не нарушались ограничения из списка запретов. Такой путь можно построить, например, следующим образом (рис. 11).

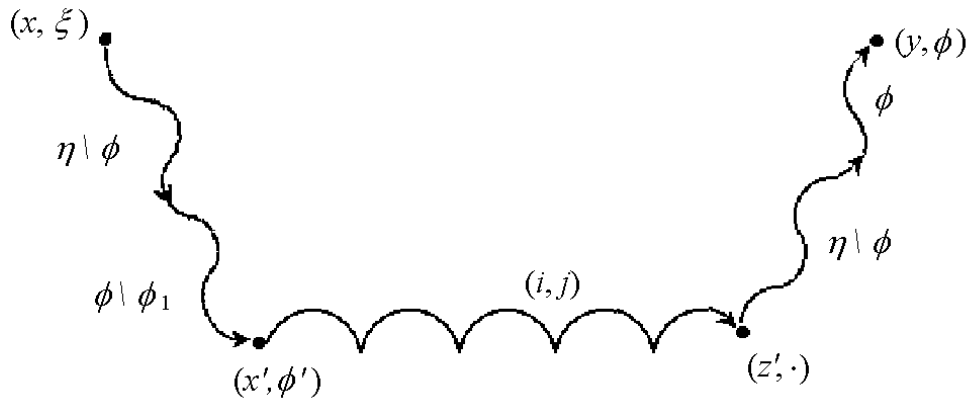


Рис. 11. *Путь от* (x, ξ) *до* (y, ϕ)

Обозначим через η множество всех пар (i, j), $i < j$, и положим $\phi_1 = \phi \setminus \eta$. На первом этапе осуществим переход из точки x в точку x' со списком запретов ϕ', удовлетворяющим условию $\eta \supseteq \phi' \supseteq \phi \setminus \phi_1$. Для достаточно этого сначала сделать l шагов по парам из $\eta \setminus \phi$, а затем не более l шагов по $\phi \setminus \phi_1$. На втором этапе, меняя на каждом шаге только одну координату, переходим по координатам из списка ϕ_1 в точку z, которая при четном n отличается от точки $1 - y$, а при нечетном n — от точки y. Наконец, на последнем этапе осуществляем переход из точки z в заданную точку y, получив в конце пути список ϕ. Для этого достаточно выполнить все шаги из множества $\eta \setminus \phi$, а затем все шаги в соответствии со списком ϕ, используя его в обратном порядке. Так как при выполнении всех шагов из η переходим из z в z при нечетном n и в $1 - z$ при четном n, то в итоге получаем требуемый путь. □

Следствие 2 *При любом* $x^0 \in E^n$ *и любых* l *и* p *таких, что* $0 < l < n(n-1)/4$ *и* $0 < p < 1$, *справедливы утверждения*:

1) $F_A \to F_{opt}$ *почти наверное при* $t \to \infty$;

2) *существуют такие константы* $b > 1$ *и* $0 < c < 1$, *что* $Pr\{F_A \neq F_{opt}\} \leqslant bc^t$;

3) *цепь Маркова* $\{(x^t, \varphi^t)\}$ *имеет единственное стационарное распределение* $\pi(x, \varphi) > 0$.

Доказательство. Первое и второе утверждения непосредственно следуют из свойства неразложимости цепи и того факта, что величина F_A монотонно не возрастает с ростом числа шагов алгоритма. Чтобы проверить третье утверждение, достаточно заметить (см. [13], гл. 12), что цепь (x^t, φ^t) является непериодической, так как при $N_k(x^t, \varphi^t, p) = \emptyset$ имеем

$x^{t+1} = x^t$. □

Первое утверждение следствия фактически означает, что алгоритм найдет оптимальное решение за достаточно большое число шагов. Нельзя утверждать, что последовательность $\{x^t\}$ сходится к оптимальному решению, как это имеет место для алгоритма имитации отжига [70]:

$$\lim_{t\to\infty} \Pr\{x^t \in X_{opt}\} = 1,$$

где X_{opt} — множество оптимальных решений. Однако для практических целей это не имеет большого значения, так как всегда можно взять лучшее решение F_A из последовательности $\{F(x^\tau)\}$, $\tau \leq t$. Вариант поиска с запретами, обладающий указанным свойством, можно найти в [115]. Следует отметить, что этот вариант использует другой способ рандомизации и концептуально близок к схеме имитации отжига.

Второе утверждение гарантирует сходимость по вероятности величины F_A к оптимальному решению задачи со скоростью геометрической прогрессии. По-видимому, в общем случае нельзя получить нетривиальные оценки на константу c, не делая никаких предположений относительно целевой функции [197, 209]. В частных случаях, учитывая специфику задачи, это удается сделать, правда, для более простых алгоритмов случайного блуждания [196].

Третье свойство гарантирует, что алгоритм порождает эргодическую цепь Маркова с положительным стационарным распределением $\pi(x, \varphi)$. Отметим, что этим свойством обладают многие вероятностные алгоритмы. Например, алгоритм, на каждом шаге которого выбирается очередная точка в E^n случайным образом с равномерным распределением. Для него $\pi(x) = 1/|E^n| = 2^{-n}$, $x \in E^n$. Поэтому сам факт существования положительного стационарного распределения не является новым. Удивительно то, насколько сильно эта величина для вероятностного поиска с запретами отличается в точке глобального оптимума от 2^{-n}. Численные эксперименты на дискретных задачах размещения дают величины порядка $10^{-3} - 10^{-4}$ при $n = 100$ на сложных классах данных. Для метрических задач той же размерности эта величина имеет порядок 10^{-2}. Таким образом, вероятностный поиск с запретами действительно имеет большие шансы найти глобальный оптимум.

3.3.3 Варианты алгоритмов

Выше предполагалось, что величина l не меняется в ходе работы алгоритма. Это создает определенные трудности при реализации схемы, так как заранее неизвестно, какой длины следует брать список запретов.

При малых l алгоритм может начать «*циклить*». При больших l поиск становится малоэффективным. Одно из простых правил регулирования длины списка запретов состоит в следующем. Если на очередном шаге алгоритма окрестность $N_k(x^t, \varphi^t, p)$ оказалась пустой, то в список запретов добавляется фиктивная пара $(0, 0)$. Эта пара ничего не запрещает, но уменьшает число запретов на единицу. Такой вариант алгоритма с саморегуляцией l также обладает указанными асимптотическими свойствами, но уже при любых $l \geq 0$.

Другое правило управления списком запретов состоит в следующем [89]. На первом шаге алгоритма полагается $l = 1$. Затем длина списка меняется в интервале от 1 до $l_{\max}$ в зависимости от того, встречалась ли точка x^{t+1} ранее на последних, скажем, L_1 шагах алгоритма. Если да, то величина l увеличивается:

$$l := \min\bigl\{l_{\max},\ \max\{l+1;\ 1.1l\}\bigr\}.$$

Если же за L_2 шагов алгоритма все точки были новыми, то величина l уменьшается:

$$l := \max\bigl\{1,\ \min\{l-1;\ 0.9l\}\bigr\}.$$

Хранение и поиск решений осуществляются при помощи хэш-функций [33], которые позволяют выполнять эти операции в среднем за $O(1)$ действий. Проверку списка запретов и его изменение на каждом шаге алгоритма тоже можно осуществлять за $O(1)$ действий. Для этого достаточно хранить список запретов одновременно в двух видах: в виде булевой матрицы и вектора, организованного в виде очереди. Проверку принадлежности списку запретов легко проводить по матрице, а изменение списка — по вектору. Если максимальная длина списка $l_{\max}$ не превышает $n(n-1)/4$ или в список запретов добавляются фиктивные пары при пустой окрестности, то такой вариант алгоритма тоже обладает асимптотическими свойствами.

Рассмотрим вариант алгоритма, который использует так называемый *критерий ускорения* (aspiration criterion [148]). Эта модификация состоит в том, чтобы отменять запреты и осуществлять переход в *запрещенную* точку y, если $F(y) < F_A$. Очевидно, что точка y ранее не встречалась,

и запрет только мешает работе алгоритма. Теперь множество состояний описывается не парой (x^t, φ^t), а тройкой (x^t, φ^t, x_A), где x_A — наилучшее решение, найденное алгоритмом к шагу t. Выбор очередной тройки снова не зависит от предыстории, и алгоритм порождает цепь Маркова.

Отметим, что при других способах рандомизации окрестности указанные асимптотические свойства могут и не иметь место; например, если брать из окрестности r случайных точек [129]. В частности, для $r = |N_k(x)| - l$ получаем детерминированный алгоритм поиска с запретами. Если список запретов слишком мал, то такой алгоритм находит локальный оптимум и не в состоянии его покинуть. Поэтому требуется достаточно большой список запретов.

Ниже приведен пример задачи минимизации, для которой детерминированный алгоритм поиска с запретами не способен найти оптимальное решение при больших списках запретов.

Таблица 6. *Поведение детерминированного алгоритма*

Циклы	Целевая функция
(1,1), (1,2), (2,2)	-3 , -3 , -4
(3,3), (1,3), (1,4), (4, 4)	-3 , -3 , -3 , -4
...	-3 , -3 , -3 , -4
$(n-1, n-1)$, $(1, n-1)$, $(1, n)$, (n, n)	-3 , -3 , -3 , -4
(2,3), (3,4), (2,4)	-2, -2, -4
(2,3), (3,4), (2,4)	-2, -2, -2, -4
...	-2, -2, -2, -4
$(2, n-l)$, $(3, n-l)$, $(3, n)$, $(2, n)$	-2, -2, -2, -4
...	...
$(n-2, n-1)$, $(n-1, n)$, $(n-2, n)$	-2, -2, -4

Пусть $F(x) = -|4 - x_1 - \ldots - x_n|$, где $n \geq 10$ и четно, а $x_i \in \{0, 1\}$. Глобальный минимум равен $4 - n$. Положим $x^0 = (0, \ldots, 0)$ и рассмотрим поведение детерминированного алгоритма для этого примера. На первом шаге список запретов пуст, т. е. $\varphi^0 = \varnothing$, и требуется найти точку x^1 такую, что $F(x^1) = \min\{F(y) \mid y \in N_2(x^0, \varphi^0)\}$. Таких точек много.

Выбираем $x^1 = (1, 0, \ldots, 0)$ как лексикографически максимальную. Пара $(1, 1)$ добавляется в список запретов, $\varphi^1 = \{(1, 1)\}$, так как изменилась только первая координата, $F(x^1) = -3$. На втором шаге получаем $x^2 = (0, 1, 0, \ldots, 0)$, $\varphi^2 = \{(1, 2), (1, 1)\}$, $F(x^2) = -3$. На третьем шаге возвращаемся к начальной точке $x^3 = x^0, \varphi^3 = \{(2, 2), (1, 2), (1, 1)\}$, $F(x^3) = -4$. Алгоритм сделал первый цикл. В дальнейшем он будет возвращаться к x^0 много раз через циклы длины 3 или 4 (табл. 6).

Легко проверить, что при $l \geq |N_2(x^0)|$ алгоритм поместит все пары $(i, j), i \leq j$, в список запретов и остановится в точке $x^{n(n+1)/2} = x^0$. Если при пустой окрестности $N_2(x^t, \varphi^t)$ алгоритм добавляет в список φ^t фиктивную пару $(0, 0)$, то через $l + 1 - |N_2(x^t, \varphi^t)|$ шагов первая пара $(1, 1)$ окажется свободной и все циклы будут повторяться снова. В противном случае $x^{t+1} = x^t$, где $t \geq n(n + 1)/2$.

При $l < |N_2(x^0)|$ алгоритм тоже делает циклы, если $l = 2 + 4s$, $s = 0, \ldots, n/2-1$; $l = 2n+1+4s$, $s = 0, \ldots, n/2-2$; и др. (см. табл. 6). Таким образом, для любого l найдется $l', |l-l'| \leq 2$, такое, что при длине списка запретов l' детерминированный алгоритм не сможет найти оптимальное решение.

3.3.4 Вычислительные эксперименты

Разработанные алгоритмы тестировались на примерах из электронной библиотеки «Дискретные задачи размещения» (http://math.nsc.ru/AP/benchmarks/). Тестовые примеры имели следующие параметры:

$$|J| = 100, \quad |I| = 50, \quad |P| = 100, \quad f_i = 3000, \quad f_p^0 = 0, \quad |I_p| = 4.$$

Матрица c_{pj} формировалась следующим образом. Для каждого $j \in J$ случайным образом определялось подмножество $P(j) \subset P$, $|P(j)| = 10$, технологических цепочек, доступных потребителю j. Затем с помощью псевдослучайной величины δ с равномерным распределением на множестве $\{0, 1, 2, 3, 4\}$ полагалось

$$c_{pj} = \begin{cases} \delta, & \text{если } p \in P(j), \\ \infty & \text{в противном случае.} \end{cases}$$

Для данного класса задач было сформировано 30 примеров. Методом ветвей и границ [19] в каждом примере найдено точное решение. Класс

задач оказался достаточно сложным, так как в нем, в частности, содержится нетривиальная подзадача о покрытии. Среднее значение разрыва целочисленности

$$\Delta = 100\%(F_{opt} - F_{LP})/F_{opt},$$

где F_{LP} — оптимальное значение линейной релаксации, для этих задач составило около 45%. Значительный разрыв двойственности предопределил большое число просматриваемых вершин в дереве ветвлений, в среднем около 10^6. Указанный класс значительно сложнее классов из [10, 19, 21] как для точных методов, так и для вероятностных жадных алгоритмов и итерационных методов локального поиска.

Для сокращения трудоемкости одного шага алгоритма рассматривалась более узкая окрестность

$$N(x) = N_1(x) \cup N'(x) \subset N_2(x),$$

где $N'(x) = \{y \in N_2(x) \mid d(y,0) = d(x,0)\}$. Все эксперименты проводились для алгоритма с этой окрестностью и с добавлением в список запретов фиктивной пары $(0,0)$, если рандомизированная окрестность оказывалась пустой.

Первый вычислительный эксперимент связан с поиском наилучших значений p для разных длин списка запретов. Обозначим через $\varepsilon(T)$ среднюю относительную погрешность

$$\varepsilon(T) = 100\%(F_A - F_{opt})/F_{opt}$$

после выполнения T шагов алгоритма. Изменение погрешности $\varepsilon(T)$ для одной из тестовых задач при $l = 0, 10, 50$ изображено на рис. 12.

Каждая точка на графике показывает среднюю погрешность за 50 испытаний. Наилучшее значение p^* лежит в интервале $[3\%, 9\%]$. Для сравнения $\varepsilon(T) = 17,4\%$ при $p = 100\%,\ l = 50$. Время счета на PC Pentium II 800 MHz составило 152 секунды. При рандомизации $p = 10\%$ время счета составило 21 секунду. Следовательно, детерминированный алгоритм поиска с запретами дает в среднем худшее решение и затрачивает на его нахождение больше времени.

Аналогичная картина наблюдается и на остальных примерах данного класса. За 20 тысяч шагов детерминированный алгоритм в 95 случаях из 100 давал большую погрешность, чем вероятностный аналог. По-видимому, такая картина должна иметь место для сложных примеров, хотя интуитивно кажется, что чем сложнее задача, тем тщательнее надо

просматривать окрестность, и всегда лучше просматривать всю окрестность, чем ее часть. Тем не менее это не так, по крайней мере для многостадийной задачи размещения. Для простых задач, когда любой локальный оптимум является глобальным, детерминированный алгоритм будет скорее всего выигрывать у вероятностного алгоритма. На сложных же задачах, в которых много локальных оптимумов и они непредсказуемым образом разбросаны по допустимой области, преимущество, как подсказывает наш опыт, должно быть на стороне вероятностных алгоритмов.

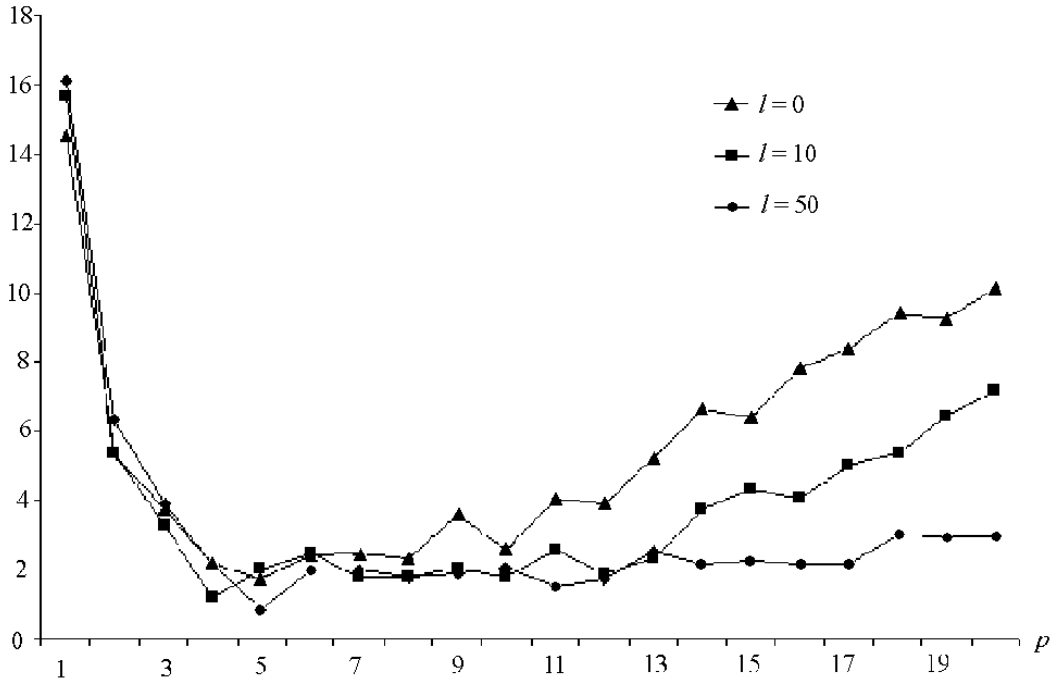


Рис. 12. *Изменение* $\varepsilon(p)$ *при* $T = 20000$

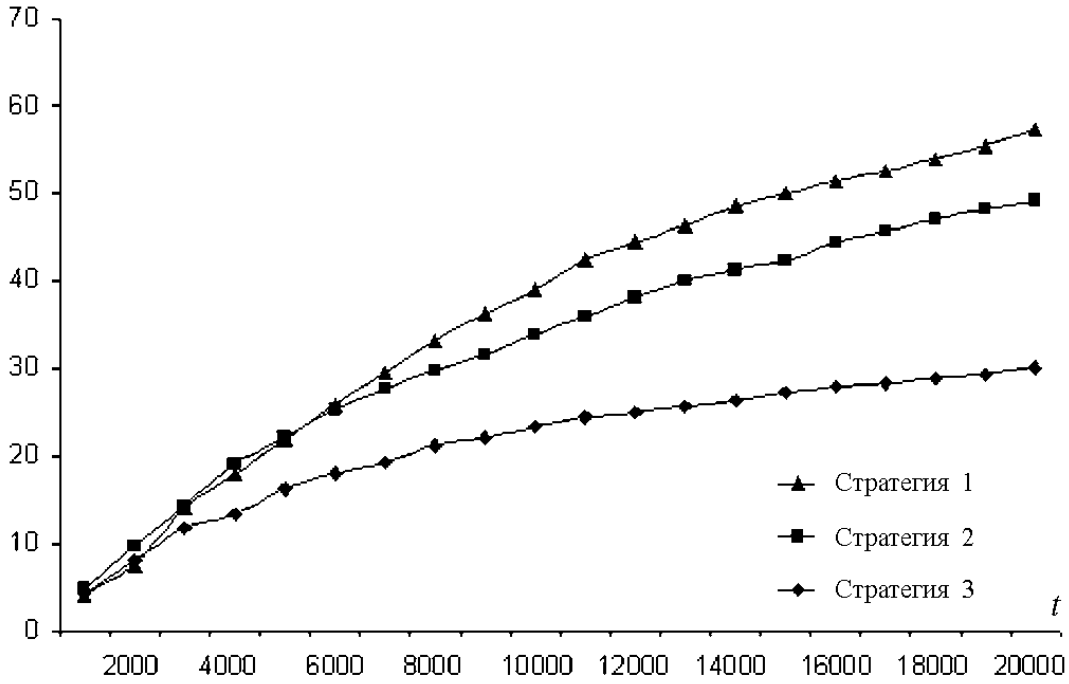


Рис. 13. *Частота получения точного решения для* $l = 50$

Заметим, что относительная погрешность невелика при $l = 0$. Другими словами, вероятностный алгоритм локального поиска без списка запретов позволяет находить достаточно хорошие решения при малых значениях p. Влияние списка запретов уменьшается при уменьшении p. Таким образом, рандомизация окрестности сокращает относительную погрешность, время счета и влияние списка запретов.

Второй вычислительный эксперимент связан с критерием ускорения (aspiration criterion) и его влиянием на относительную погрешность. Напомним, что согласно этому критерию разрешается игнорировать список запретов, если в окрестности текущей точки найдено лучшее значение целевой функции, чем F_A. Интуитивно понятно, что такой вариант алгоритма может иметь определенные преимущества, однако численные эксперименты на указанном классе исходных данных не подтверждают это предположение. По-видимому, лучшее значение оказывается запрещенным крайне редко, либо алгоритм многократно возвращается к нему с разными списками запретов и в итоге все равно находит эту точку или равноценную ей.

Третий вычислительный эксперимент связан с тезисом о *большой долине* [34, 94]. Согласно этому тезису в среднем локальные оптимумы располагаются значительно ближе к глобальному оптимуму, чем случайно выбранная точка. Распределение локальных оптимумов не явля-

ется равномерным. Существует определенная концентрация локальных оптимумов в малой части допустимой области, которую образно называют *горным массивом* (massive central) для задач на максимум или *большой долиной* (big valley) для задач на минимум. Если это предположение верно, то при решении задачи целесообразно запоминать разные точки с небольшим отклонением от значения F_A, т. е. формировать *элиту*, и время от времени выбирать новую стартовую точку среди точек элиты.

Это правило лежит в основе оператора кроссовера (crossover) для генетических алгоритмов [187]. Точнее, новая стратегия выбора стартовой точки заключается в следующем. В ходе работы алгоритма формируется *популяция* решений, для которых значение целевой функции незначительно отличается от значения F_A. Это множество строится так, чтобы все решения были различны и расстояние Хемминга между ними было достаточно велико. В ходе численных экспериментов это расстояние было не меньше 5, размер популяции равен 10. При смене стартовой точки из этого множества равновероятно выбирались два решения (родители). На их основе строилось новое решение, каждая компонента которого с равной вероятностью принимала значение соответствующей компоненты одного из родителей (uniform crossover). Затем к полученному решению применялся вероятностный оператор *мутации*, который с заданной вероятностью (0,05) менял значение каждой компоненты на противоположное $x_i \to 1 - x_i$. В дальнейшем такую стратегию выбора стартовой точки будем называть генетическим стартом.

На рис. 13 изображена частота нахождения оптимального решения для следующих трех стратегий:

1) генетический старт;

2) выбор случайной стартовой точки, если рекорд F_A не меняется в течение 500 шагов;

3) стандартная стратегия без смены стартовой точки.

Частота получения с помощью указанных стратегий приближенного решения с погрешностью не более 1% изображена на рис. 14. Каждая точка на этих графиках показывает среднюю величину по 30 испытаниям в 30 тестовых примерах.

В табл. 7 приведены доверительные интервалы для вероятности получения за 20 тысяч шагов алгоритма точного решения и приближенного решения с погрешностью не более 1%. Коэффициент доверия 0,95.

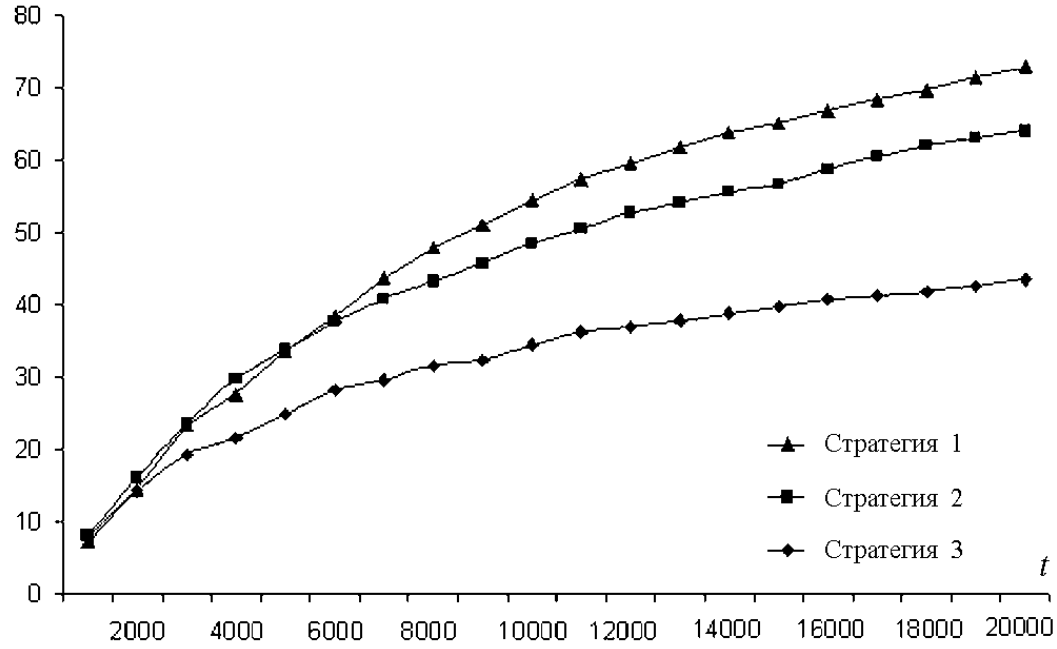


Рис. 14 *Частота получения решений с погрешностью* $\varepsilon \leqslant 1\%; l = 50$

Рис. 13 и 14 свидетельствуют о преимуществе стратегии, основанной на гипотезе большой долины. По-видимому, данный эксперимент можно рассматривать как косвенное подтверждение гипотезы о концентрации локальных оптимумов для исследуемого класса исходных данных.

Таблица 7. *Доверительные интервалы*

Стратегия	$\varepsilon = 0$	$\varepsilon \leq 1\%$
1	$0,541 - 0,605$	$0,701 - 0,759$
2	$0,460 - 0,524$	$0,610 - 0,672$
3	$0,272 - 0,330$	$0,403 - 0,467$

3.3.5 Направления дальнейших исследований

Выше рассматривались варианты алгоритма, когда порог рандомизации окрестности был фиксирован. Увеличение порога ведет к более тщательному исследованию области решений, близких к x^t. При уменьшении порога алгоритм стремится сменить область поиска. Управление этим параметром может повысить эффективность алгоритма и решить проблему выбора оптимальной рандомизации поиска. Разработка адаптивной

стратегии управления этим параметром представляется важным направлением на пути совершенствования данного подхода.

Другим интересным направлением исследований является поиск наиболее трудных классов задач для локального поиска. Если число локальных оптимумов растет экспоненциально с ростом размерности задачи [158], то для локального поиска задача становится достаточно трудной. Интересно найти другие трудные случаи и дать их классификацию, например, с точки зрения распределения локальных оптимумов. Генерация трудных примеров и объяснение, почему эти примеры оказываются трудными для разных подходов, способствуют лучшему пониманию природы NP-трудных задач и совершенствованию алгоритмов их решения.

3.4 Генетический локальный поиск

Эволюционные алгоритмы (ЭА), к числу которых относятся генетические алгоритмы [149], эволюционные стратегии [184], алгоритмы генетического программирования [162] и др., берут свое начало в пионерских работах Дж. Голланда [149], Л. Фогеля, А.Оуэнса и М. Уолша [65]. В этих работах принципы биологической эволюции были впервые использованы для решения задач синтеза эффективных структур и создания систем искусственного интеллекта. Почти одновременно с возникновением метода эволюционного моделирования А.Г. Ивахненко и Л.А.Растригиным были предложены методы группового учета аргументов [30] и случайного поиска [54, 53], где также использовались идеи эволюции и случайной мутации.

Основная идея ЭА состоит в компьютерном моделировании процесса эволюции, основанном на факторах изменчивости, наследования и отбора наиболее приспособленных особей. Эволюционное моделирование используется не для исследования биологических популяций, а для решения задач оптимизации и принятия решений. Разработчик ЭА выступает в данном случае как "создатель", который должен правильно установить законы эволюции, чтобы достичь желаемой цели, т.е. получить оптимальное решение в сложной экстремальной задаче. Впервые эти нестандартные идеи были применены к решению оптимизационных задач в середине 70-х годов [54, 97]. Примерно через десять лет появились первые теоретические обоснования этого подхода [149, 184, 198]. На сегодняшний день генетические алгоритмы доказали свою конкурентоспособность при решении многих NP-трудных задач [27, 131, 185, 186, 105, 171] и

особенно в практических приложениях, где математические модели имеют сложную структуру и применение стандартных методов типа ветвей и границ, динамического или линейного программирования крайне затруднено [211, 98].

Одним из наиболее эффективных вариантов ЭА является алгоритм генетического локального поиска. В этом алгоритме элементами популяции являются локальные оптимумы. В англоязычной литературе такие алгоритмы получили название Memetic Algorithms (MA). Интерес к ним с математической точки зрения вполне оправдан. Если элементами популяции являются только локальные оптимумы, то в ходе работы алгоритма находятся все новые и новые локальные оптимумы. Возникают естественные вопросы о качестве локальных оптимумов, трудоемкости их получения, характеристиках ландшафтов и др. Таким образом, MA можно рассматривать как усиленный вариант ЭА, в котором основной упор делается на целенаправленный поиск глобального оптимума или локальных оптимумов с малой погрешностью.

3.4.1 Генетический алгоритм

Общую схему генетических алгоритмов проще всего понять, рассматривая задачи безусловной оптимизации

$$\max\{F(s) \mid s \in B^n\}, \quad B^n = \{0,1\}^n.$$

Примерами таких задач служат задачи размещения, стандартизации, выполнимости и др. Стандартный генетический алгоритм начинает свою работу с формирования начальной *популяции* $\Pi_0 = \{s_1, s_2, \ldots, s_m\}$ — конечного набора допустимых решений задачи. Эти решения могут быть выбраны случайным образом или получены с помощью вероятностных жадных алгоритмов. Как мы увидим ниже, выбор начальной популяции не имеет значения для сходимости процесса в асимптотике, однако формирование "хорошей" начальной популяции (например, из множества локальных оптимумов) может заметно сократить время достижения глобального оптимума.

На каждом шаге эволюции с помощью вероятностного оператора *селекции* выбираются два решения, *родители* s_1, s_2. Оператор *скрещивания* по этим решениям строит новое решение s'. Оно подвергается небольшим случайным модификациям, которые принято называть *мутациями*. Затем это решение добавляется в популяцию, а решение с наименьшим значением целевой функции удаляется из популяции. Общая схема

такого алгоритма может быть записана следующим образом.

Генетический алгоритм

1. Выбрать начальную популяцию Π_0 и положить $k = 0$.
2. Пока не выполнен критерий остановки делать следующее.
   2.1. (Селекция) Выбрать родителей s_1, s_2 из популяции Π_k.
   2.2. (Скрещивание) Построить s' по решениям s_1, s_2.
   2.3. (Мутация) Модифицировать s'.
   2.4. Обновить популяцию и положить $k = k + 1$.
3. Предъявить лучшее найденное решение.

Остановимся подробнее на основных операторах этого алгоритма: селекции, скрещивании и мутации. Среди операторов селекции наиболее распространенными являются два вероятностных оператора *пропорциональной* и *турнирной* селекции. При пропорциональной селекции вероятность на k-м шаге выбрать решение s_i в качестве одного из родителей задается формулой

$$\mathbb{P}\{s_i\} = \frac{F(s_i)}{\sum\limits_{s \in \Pi_k} F(s)}, \ s_i \in \Pi_k,$$

в предположении, что $F(s) > 0$ для всех $s \in Sol$. При турнирной селекции формируется случайное подмножество из элементов популяции и среди них выбирается один элемент с наибольшим значением целевой функции. Турнирная селекция имеет определенные преимущества перед пропорциональной, так как не теряет своей избирательности, когда в ходе эволюции все элементы популяции становятся примерно равными по значению целевой функции. Операторы селекции строятся таким образом, чтобы с ненулевой вероятностью любой элемент популяции мог бы быть выбран в качестве одного из родителей. Более того, допускается ситуация, когда оба родителя представлены одним и тем же элементом популяции.

Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*). Существует много различных версий этого оператора [131], среди которых простейшим, по видимому, является однородный оператор. По решениям s_1, s_2 он строит решение s' присваивая каждой его координате значение соответствующей координаты одного из родителей с вероятностью 0,5. Если векторы s_1, s_2 совпадают,

скажем, по первой координате, то вектор s' "унаследует" это значение. Геометрически, оператор скрещивания случайным образом выбирает в гиперкубе вершину s', которая принадлежит минимальной грани, содержащей вершины s_1, s_2. Можно сказать, что оператор скрещивания старается выбрать новое решение s' где-то между s_1 и s_2, полагаясь на удачу. Более аккуратная процедура могла бы выглядеть таким образом. Новым решением s' является оптимальное решение исходной задачи на соответствующей грани гиперкуба. Конечно, если расстояние Хэмминга между s_1 и s_2 равно n, то задача оптимального скрещивания совпадает с исходной. Тем не менее, даже приближенное решение этой задачи вместо случайного выбора заметно улучшает работу генетического алгоритма [72, 86, 27]. По аналогии с однородным оператором скрещивания легко предложить и другие операторы, использующие не только два, но и произвольное число решений из популяции. Например, в [175] использовалось восемь родителей. Другие примеры можно найти в [105, 98].

Оператор мутации, применяемый к решению s' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0, 1)$ меняет значение каждой координаты на противоположное. Например, вероятность того, что $s' = (0, 0, 0, 0, 0)$ в ходе мутации перейдет в $s = (1, 1, 1, 0, 0)$, равна $p_m \times p_m \times p_m \times (1-p_m) \times (1-p_m) > 0$. Таким образом, с ненулевой вероятностью решение s' может перейти в любое другое решение. Отметим, что модификация решения s' может состоять не только в случайной мутации, но и в частичной перестройке решения алгоритмами локального поиска. Применение локального спуска позволяет генетическому алгоритму сосредоточиться только на локальных оптимумах. Множество локальных оптимумов может оказаться экспоненциально большим и на первый взгляд кажется, что такой вариант алгоритма не будет иметь больших преимуществ. Однако экспериментальные исследования распределения локальных оптимумов свидетельствуют о высокой концентрации их в непосредственной близости от глобального оптимума [94]. Отчасти этот тезис объясняет работоспособность генетических алгоритмов. Если в популяции собираются локальные оптимумы, которые сконцентрированы в одном месте, и очередное решение s' выбирается где-то между двумя произвольными локальными оптимумами, то такой процесс имеет много шансов найти глобальный оптимум. Аналогичные рассуждения объясняют работоспособность и других локальных алгоритмов. Более того, этот тезис подсказывает способ построения сложных тестовых примеров. Если локальные оптимумы удается распределить по всей допустимой обла-

сти и нет концентрации хороших локальных оптимумов в малой части допустимой области, то методам локального поиска будет трудно найти точное решение задачи. Примеры таких классов исходных данных для задач размещения можно найти, например, в [157, 158].

Перейдем теперь к анализу генетического алгоритма с позиций цепей Маркова [191]. На каждом шаге эволюции алгоритм имеет дело с популяцией — набором из m случайных решений. Множество состояний такого стохастического процесса — все возможные популяции. Переход одной популяции к другой не зависит от того, каким образом была получена данная популяция, а зависит только от состава самой популяции. Другими словами, генетический алгоритм порождает цепь Маркова на конечном множестве состояний. Так как оператор мутации за один шаг позволяет достичь любое решение задачи и, в частности, оптимальное решение, то с ростом числа шагов вероятность найти глобальный оптимум стремится к единице. Если при изменении популяции наилучшее найденное решение всегда остается в популяции, то в пределе она будет состоять из одних оптимальных решений задачи. Таким образом, имеет место сходимость по вероятности наилучшего найденного решения к глобальному оптимуму. Отметим, что аналогичным свойством обладает и примитивный случайный поиск. Поэтому данное утверждение говорит скорее о правилах построения мутаций. Их нужно конструировать таким образом, чтобы имелась ненулевая вероятность перехода в произвольное решение задачи. Отсутствие оценок скорости сходимости свидетельствует о слишком большой общности рассматриваемого алгоритма. Для получения нетривиальных оценок нужно конкретизировать решаемую задачу, учесть ее специфику. Без этих уточнений в самом общем виде усилить результаты об асимптотическом поведении ЭА вряд ли удастся.

3.4.2 Генетический алгоритм для задачи МПК

Рассмотрим варианты адаптации общей схемы генетических алгоритмов к задаче о p-медиане с предпочтениями клиентов (МПК) из раздела 1.3. Напомним, что эта задача может быть представлена следующим образом: найти

$$\min_{y} F(y)$$

$$\sum_{i \in I} y_i = p$$

$$y_i \in \{0,1\}, i \in I.$$

В качестве пространства поиска естественно взять p-й слой гиперкуба B^m, т. е. все векторы $y_i \in \{0,1\}, i \in I$, удовлетворяющие условию $\sum_{i\in I} y_i = p$, а в качестве функции пригодности — целевую функцию $F(y)$. Такой выбор представляется обоснованным, так как целевая функция чувствительна к небольшим изменениям аргумента.

Операторы селекции используют функцию $F(y)$ для выбора родительской пары. Наряду с турнирной и пропорциональной селекцией рассмотрим третий оператор, получивший название "лучший и случайный". Согласно этому оператору в качестве одного из родителей всегда берется лучший элемент популяции. Второй родитель выбирается из оставшихся элементов случайным образом с равномерным распределением. Такой оператор позволяет имитировать поведение алгоритма локального поиска с чередующимися окрестностями [170], когда хранится только наилучший найденный локальный оптимум.

После выбора родительской пары y^1, y^2 выполняется оператор скрещивания. Рассмотрим четыре типа таких операторов: равномерный, жадный, одноточечный и связывающих путей (path relinking).

При **равномерном** операторе новое допустимое решение y' строится следующим образом. Если $y_i^1 = y_i^2$ для некоторого $i \in I$, то y'_i наследует это значение: $y'_i = y_i^1 = y_i^2$. Для остальных $i \in I$ сначала полагается $y'_i = 0$, а затем выполняется следующая итерационная процедура. Выбирается одна из координат, для которой $y'_i = 0$ и полагается $y'_i = 1$. Выбор координаты проводится случайным образом с равномерным распределением. Процедура останавливается, как только $\sum_{i\in I} y'_i = p$.

При **жадном** операторе итерационная процедура предусматривает более осмысленный выбор координаты, для которой y'_i полагается равной единице. Среди претендентов выбирается та координата, для которой уменьшение целевой функции будет наибольшим. Если уменьшение целевой функции таким способом невозможно, то выбирается координата с наименьшим увеличением целевой функции. В остальном жадный оператор совпадает с равномерным. По сути данный оператор предполагает решение исходной задачи с помощью жадной процедуры на множестве координат, где родители отличаются друг от друга.

Одноточечный оператор скрещивания хорошо известен и часто применяется в генетических алгоритмах [149]. Согласно правилам этого оператора выбирается одна координата, скажем i_0, и полагается $y'_i = y_i^1$ для $i \le i_0$ и $y'_i = y_i^2$ для $i > i_0$. Координата i_0 выбирается случайным

образом, например, с равномерным распределением. Для задачи МПК такой способ может приводить к недопустимым решениям, $\sum_{i \in I} y'_i \neq p$, поэтому требуется модификация этого оператора. Пусть как и раньше $y'_i = y^1_i$ для $i \leq i_0$. Если $\sum_{i \leq i_0} y'_i < p$, то среди единичных компонент вектора $y^2_i, i > i_0$, выберем первые $(p - \sum_{i \leq i_0} y'_i)$ компонент и положим для них $y'_i = 1$. Остальные компоненты положим равными 0. Если же вектор $y^2_i, i > i_0$ не содержит достаточное число единиц, то аналогичным образом используются компоненты вектора $y^2_i, y^2_i \neq y^1_i, i \leq i_0$. Заметим, что так построенное решение имеет как минимум два недостатка. Единичные компоненты вектора y^2 с большими номерами имеют мало шансов быть унаследованными в векторе y'. Кроме того, вектор y' может отличаться от родителей по тем координатам, где $y^1_i = y^2_i$. Тем не менее, эти недостатки, как мы увидим ниже, сглаживаются при правильном выборе параметров алгоритма и эффективном локальном поиске.

Наконец, оператор **связывающих путей** строит вектор y' следующим образом. Из двух решений y^1 и y^2 выбирается решение с меньшим значением целевой функции и строится последовательность допустимых решений $\overline{y}^1, \overline{y}^2, \ldots, \overline{y}^k$, таких, что $\overline{y}^1 = y^1, \overline{y}^k = y^2$ и расстояние Хэмминга между соседними решениями равно двум. Другими словами, переход к соседнему решению осуществляется за счет закрытия одного предприятия и открытия другого. Решение y' выбирается среди элементов этой последовательности. Для заданных y^1, y^2 можно построить экспоненциально много таких последовательностей. Чтобы решение y' унаследовало общие компоненты родителей, следует рассматривать только кратчайшие последовательности. Однако, таких тоже много. Среди кратчайших последовательностей принято выбирать ту, в которой переход к соседнему решению сопровождается наибольшим уменьшением (или наименьшим увеличением) целевой функции. Это аналог жадной процедуры, но основанной на других принципах. Решение y' выбирается в этой последовательности таким образом, чтобы соседние с ним решения не имели меньших значений целевой функции [185]. Если такой выбор невозможен, то в качестве y' берется решение из середины последовательности.

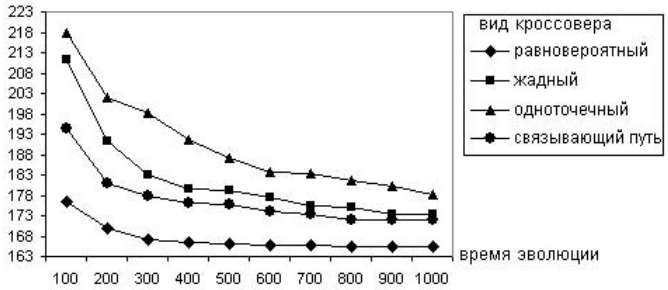
Оператор **мутации** случайным образом преобразует решение y' в другое допустимое решение y''. С заданной вероятностью каждая единичная компонента вектора y' становится равной 0, а одна из нулевых компонент, выбранная случайным образом с равномерным распределе-

нием, получает значение 1.

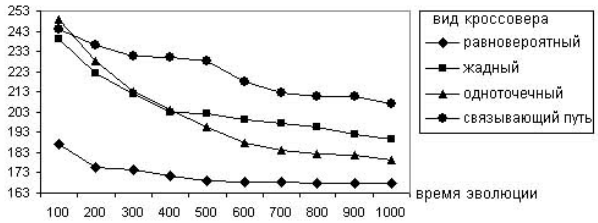
К решению y'' применяется процедура локального спуска сначала по окрестности N_1, а затем по окрестности Лина-Кернигана [157]. Полученный локальный оптимум добавляется в популяцию, если он там еще не содержится, и наихудшее решение удаляется из популяции. Критерием остановки генетического алгоритма служит число полученных локальных оптимумов.

3.4.3 Тестовые примеры

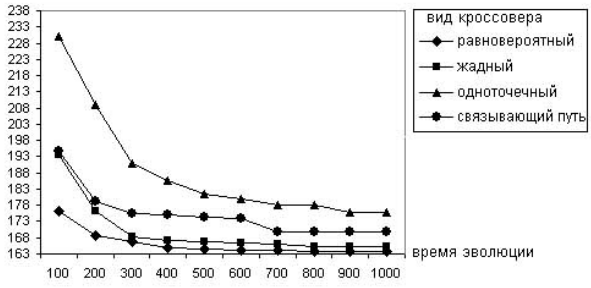
Разработанный генетический алгоритм тестировался на примерах из электронной библиотеки "Дискретные задачи размещения" (http://math.-nsc.ru/AP/benchmarks/). Размерность примеров составляла $n = m = 100$. Матрица (c_{ij}) порождалась случайным образом (класс GAP-C, [157]) и имела низкую плотность: ровно 10 элементов в каждом столбце и каждой строке имели целочисленные значения от 0 до 4. Остальные элементы полагались равными достаточно большому числу. Матрица приоритетов (g_{ij}) формировалась в два этапа. Сначала каждый столбец матрицы (c_{ij}) упорядочивался по неубыванию $c_{i_1 j} \le c_{i_2 j} \le \ldots \le c_{i_m j}$ и полагалось $g_{i_k j} = k, k = 1, \ldots m$. Затем в каждом столбце матрицы (g_{ij}) среди элементов со значением от 0 до 4 выбирались случайным образом три пары элементов и их значения менялись друг на друга. Число p полагалось равным 14. Это минимальное число, при котором все еще удается обслужить всех клиентов, не используя большие значения матрицы (c_{ij}). Нахождение оптимального решения задачи с помощью коммерческого программного обеспечения GAMS оказалось трудоемким: после 270 часов работы пакета на PC Pentium IV разрыв двойственности составил 18% и оптимальное решение всё еще не было получено. В качестве оптимизационной процедуры в GAMS использовался пакет CPLEX 6.0.



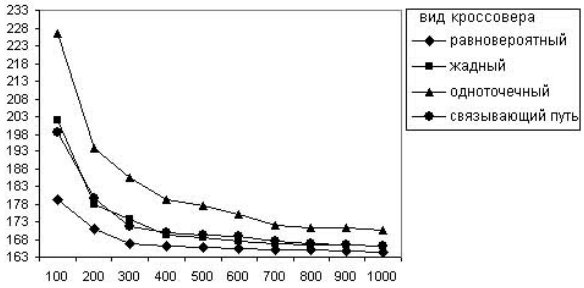
а) турнирная селекция, $p_m = 0,02$



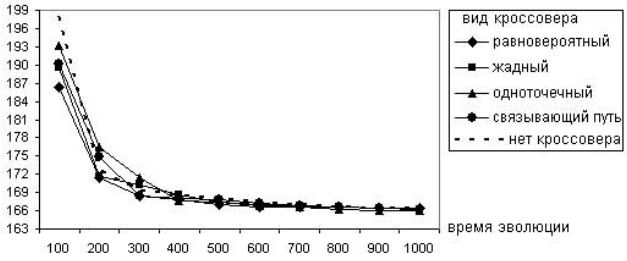
б) селекция "лучший и случайный", $p_m = 0,02$



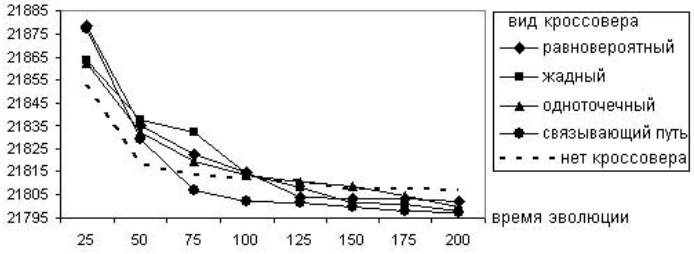
в) равновероятная селекция, $p_m = 0,02$



г) турнирная селекция, $p_m = 0,1$



д) турнирная селекция, $p_m = 0,5$



е) турнирная селекция, $p_m = 0,5$, класс Euclidean

Рис.15. *Среднее значение наилучшего решения в популяции*

3.4.4 Выбор параметров алгоритма

Результаты работы генетического алгоритма существенно зависят от его параметров и применяемых операторов селекции, скрещивания и мутации. Для повышения эффективности алгоритма был проведен следующий эксперимент. При заданном числе T получаемых локальных оптимумов проводилось сравнение лучших элементов в популяции для разных способов селекции и скрещивания. На рис. 15*а* показано изменение этих значений в ходе эволюции при турнирной селекции с вероятностью мутации, равной 0.02. Каждая точка на графике показывает среднее значение по 10 примерам и 100 испытаниям генетического алгоритма для каждого примера. Численность популяции составляла 20 локальных оптимумов. В качестве родителя всякий раз выбирался лучший по значению целевой функции из трех элементов популяции, выбранный случайным образом с равномерным распределением. Наихудшие результаты показал, как и ожидалось, одноточечный оператор скрещивания. Его отставание от других весьма велико и объясняется, по-видимому, неудачностью самой конструкции. Неожиданно хорошие результаты показал самый простой, равномерный оператор, который доминирует во всех экспериментах на данном классе. Его успех, видимо, объясняется необычайной сложностью самого класса тестовых примеров, в которых нахождение глобального оптимума или хорошего приближения представляется весьма сложным делом. Жадные стратегии или стратегии связывающих путей, показавшие хорошие результаты на других задачах [129], здесь явно проигрывают.

На рис. 15*б* и 15*в* представлены результаты расчетов для двух других способов селекции. Стратегия "лучший и случайный" проигрыва-

ет двум другим, а равновероятная селекция дает, как и равновероятное скрещивание, наилучшие результаты. Интересно отметить, что повышение вероятности мутации (рис. 15*г*, 15*д*) приводит к сглаживанию различий между операторами скрещивания, и даже отсутствие скрещивания (рис. 15*д*, 15*е*) при наличии высокой вероятности мутации приводит к хорошим результатам. Тем не менее, удаление из алгоритма операторов скрещивания не является оправданным (рис. 15*е*). Для примеров Резенде и Вернека большой размерности, $n = m = 500$, $p = 50$ [185], генетический алгоритм с любым из рассмотренных операторов скрещивания дает лучшие результаты, чем без них.

3.4.5 Нижние оценки оптимума

Рассмотрим различные способы сведéния задачи МПК к задачам ЦЛП и покажем их различия при вычислении нижних оценок методами линейного программирования.

Сведéния к задачам ЦЛП

Для каждого столбца матрицы (g_{ij}) определим перестановку $(i_1^j, ..., i_m^j)$ такую, что

$$g_{i_1 j} < g_{i_2 j} < \cdots < g_{i_m j}, \tag{3.1}$$

и множества $S_{ij} = \{k \in I \mid g_{kj} < g_{ij}\}$, $T_{ij} = \{k \in I \mid g_{kj} > g_{ij}\}$, $i \in I$. Заметим, что для оптимального решения задачи клиентов имеет место следующая импликация:

$$(x_{ij} = 1) \Longrightarrow (y_k = 0),\ k \in S_{ij}.$$

Тогда задача МПК может быть представлена в виде: найти

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{3.2}$$

при ограничениях:

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{3.3}$$

$$\sum_{i \in I} y_i = p, \tag{3.4}$$

$$y_k \leq 1 - x_{ij}, \quad k \in S_{ij}, i \in I, j \in J \tag{3.5}$$

$$x_{ij} \leq y_i,\ i \in I, j \in J, \tag{3.6}$$

$$x_{ij}, y_i \in \{0,1\},\ i \in I, j \in J. \tag{3.7}$$

Действительно, для оптимального решения задачи (3.2)–(3.7) все ограничения исходной задачи будут выполнены, а группа ограничений (3.5) гарантирует, что x_{ij} будет оптимальным решением задачи клиентов. Отбрасывая условия булевости переменных, получаем линейную релаксацию задачи. Обозначим через LB_1 её оптимальное значение. Эта величина, очевидно, даёт нижнюю оценку оптимума исходной задачи. При таком сведе́нии число переменных остается прежним и равным $m + mn$, а число ограничений становится на $O(m^2 n)$ больше. Чтобы избежать большого числа дополнительных ограничений, условие (3.5) можно переписать в виде:

$$\sum_{k \in S_{ij}} y_k \leq |S_{ij}|(1 - x_{ij}), \quad i \in I, j \in J. \tag{3.8}$$

Эти ограничения получаются суммированием ограничений (3.5). Новое сведе́ние приводит аналогичным образом к новой нижней оценке. Обозначим её через LB_2. Число дополнительных ограничений сократилось до mn, оптимальное решение задачи от этого не изменилось, но линейная релаксация стала более слабой, т. е. $LB_1 \geq LB_2$. Оптимальное решение не изменится, если условие (3.5) заменить на следующее условие:

$$y_i \leq x_{ij} + \sum_{k \in S_{ij}} y_k, \quad i \in I, j \in J, \tag{3.9}$$

что даёт ещё одно сведе́ние и нижнюю оценку LB_3. Приведённые нижние оценки были предложены в [23].

Рассмотрим новое сведе́ние к задаче ЦЛП. Учитывая ограничения (3.3), (3.6), легко показать, что условие (3.5) можно заменить на условие:

$$y_i \leq x_{ij} + \sum_{k \in S_{ij}} x_{kj}, \quad i \in I, j \in J. \tag{3.10}$$

Покажем, что получаемая с помощью (3.10) новая нижняя оценка, обозначим её через LB_4, доминирует три предшествующие.

Теорема 14 $LB_4 \geq \max(LB_1, LB_2, LB_3)$.

Доказательство. Ограничение (3.3) перепишем в виде

$$1 - \sum_{k \in T_{ij}} x_{kj} = x_{ij} + \sum_{k \in S_{ij}} x_{kj}.$$

Тогда ограничение (3.10) примет вид $y_i \le 1 - \sum\limits_{k \in T_{ij}} x_{kj} \le 1 - x_{kj}$, $k \in T_{ij}$, откуда следует (3.5). Так как $x_{ij} \le y_i$ для всех $i \in I, j \in J$, то

$$x_{ij} + \sum_{k \in S_{ij}} x_{kj} \le x_{ij} + \sum_{k \in S_{ij}} y_k.$$

Следовательно, условие (3.9) выполняется. Таким образом, значение LB_4 не может быть меньше каждой из оценок LB_1, LB_2, LB_3. □

Сведе́ние к задаче о паре матриц

Рассмотрим матрицы $A = (a_{ij}), i \in I, j \in J_1$, и $B = (b_{ij}), i \in I, j \in J_2$, с одинаковым числом строк и, возможно, различным числом столбцов. Задача о паре матриц заключается в том, чтобы найти непустое подмножество строк $S \subseteq I$, на котором достигается минимум следующей целевой функции [10]:

$$R(S) = \sum_{j \in J_1} \max_{i \in S} a_{ij} + \sum_{j \in J_2} \min_{i \in S} b_{ij}.$$

Заметим, что если $J_1 = I$, $a_{ij} = a_i$ при $i = j$ и $a_{ij} = 0$ при $i \ne j$, то получаем простейшую задачу размещения [24]. Таким образом, задача о паре матриц является NP-трудной в сильном смысле.

Известно [23, 24], что для задачи МПК можно построить сведе́ние к задаче о паре матриц с дополнительным ограничением $|S| = p$. Представим матрицу (c_{ij}) в виде суммы двух матриц $c_{ij} = a_{ij} + b_{ij}$, $\quad i \in I, j \in J$, где

$$a_{i_k j} = \sum_{l=1}^{k-1} \min\{0; c_{i_{l+1} j} - c_{i_l j}\}, k = 1, \ldots, m, j \in J, \tag{3.11}$$

$$b_{i_k j} = c_{i_1 j} + \sum_{l=1}^{k-1} \max\{0; c_{i_{l+1} j} - c_{i_l j}\}, k = 1, \ldots, m, j \in J, \tag{3.12}$$

и перестановка $(i_1^j, \ldots, i_m^j), j \in J$, соответствует (3.1). Заметим, что

$$a_{i_1 j} \ge a_{i_2 j} \ge \cdots \ge a_{i_m j}, \quad b_{i_1 j} \le b_{i_2 j} \le \cdots \le b_{i_m j}$$

при всех $j \in J$. Теперь задача МПК может быть записана в следующем виде: найти

$$\min_{S \subset I, |S| = p} \sum_{j \in J} \max_{i \in S} a_{ij} + \sum_{j \in J} \min_{i \in S} b_{ij}.$$

Для получения нижней оценки эту задачу можно переписать в терминах ЦЛП и вычислить оптимум в линейной релаксации. Однако представление в терминах ЦЛП может оказаться неединственным. Как мы уже видели, разные представления могут приводить к разным нижним оценкам. Ниже будут рассматриваться два таких представления для задачи о паре матриц, первое из которых приводит к слабой нижней оценке, а второе — к нижней оценке, которая не хуже, чем L_4. Введём вспомогательные переменные $z_{ij} \in \{0,1\}, i \in I, j \in J$, и запишем задачу о паре матриц в следующем виде: найти

$$\min \left\{ \sum_{i \in I} \sum_{j \in J} a_{ij} z_{ij} + \sum_{i \in I} \sum_{j \in J} b_{ij} x_{ij} \right\} \tag{3.13}$$

при ограничениях:

$$y_i \le \sum_{k \in S_{ij}} z_{kj} + z_{ij}, \tag{3.14}$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{3.15}$$

$$\sum_{i \in I} z_{ij} = 1, \quad j \in J, \tag{3.16}$$

$$\sum_{i \in I} y_i = p, \tag{3.17}$$

$$0 \le x_{ij} \le y_i, \tag{3.18}$$

$$0 \le z_{ij} \le y_i, \tag{3.19}$$

$$z_{ij}, x_{ij}, y_i \in \{0,1\}, \tag{3.20}$$

где $i \in I, j \in J$. Так как $b_{i_1 j} \le b_{i_2 j} \le \cdots \le b_{i_m j}, j \in J$, то j-й столбец матрицы (b_{ij}) задаёт тот же порядок, что и j-й столбец матрицы (g_{ij}). Ограничение (3.14) гарантирует, что j-й клиент делает выбор согласно собственным предпочтениям. Обозначим через LB_5 оптимальное значение линейной релаксации этой задачи, а через LB_5' оптимальное значение линейной релаксации этой же задачи с дополнительными ограничениями $x_{ij} = z_{ij}, i \in I$, $j \in J$. Заметим, что $LB_5' \ge LB_5$ и $LB_5' = LB_4$.

Теорема 15 *Для любого $N > 0$ существуют исходные данные задачи МПК такие, что $LB_5 \le -N$.*

Доказательство. При $k \geq N+2$ полагаем $I = \{1, \ldots, k+p\}$, $J = \{1, 2\}$, $p \in Z^+$, $d_{ij} = k + p - i$ и

$$c_{ij} = \begin{cases} 0, & \text{если} \quad 1 \leq i < p, & j = 1; \\ k, & \text{если} \quad i = p, & j = 1; \\ 1, & \text{если} \quad p+1 \leq i \leq k+p, & j = 1; \\ 1, & \text{если} \quad 1 \leq i < p, & j = 2; \\ 0, & \text{если} \quad p \leq i \leq k+p, & j = 2. \end{cases}$$

Согласно (3.11) и (3.12) получаем

$$a_{ij} = \begin{cases} -k, & \text{если} \quad i < p, \quad j = 1; \\ 0, & \text{если} \quad i \geq p, \quad j = 1; \\ 0, & \text{если} \quad j = 2; \end{cases} \quad b_{ij} = \begin{cases} k, & \text{если} \quad i \leq p, \quad j = 1; \\ 1, & \text{если} \quad i > p, \quad j = 1; \\ c_{ij}, & \text{если} \quad j = 2; \end{cases}$$

Решение

$$x_{ij} = \begin{cases} 0, & \text{если} \quad i \leq p, \quad j \in J; \\ 1/k, & \text{если} \quad i > p, \quad j \in J; \end{cases} \quad y_i = \begin{cases} 1, & \text{если} \quad i < p; \\ 0, & \text{если} \quad i = p; \\ 1/k, & \text{если} \quad i > p; \end{cases}$$

$$z_{ij} = \begin{cases} 1 - 1/k, & \text{если} \quad i = p-1, \quad j = 1; \\ 1/k, & \text{если} \quad i = p+k, \quad j = 1; \\ 0, & \text{если} \quad i \neq p-1, \quad i \neq p+k, \quad j = 1; \\ 0, & \text{если} \quad i \leq p, \quad j = 2; \\ 1/k, & \text{если} \quad i > p, \quad j = 2; \end{cases}$$

является допустимым решением релаксированной задачи и

$$LB_5 \leq (-k)(1 - 1/k) + \sum_{i=1}^{k} 1/k = 2 - k.$$

□

Новое сведéние к задаче о паре матриц

Представим новую формулировку задачи о паре матриц в виде задачи ЦЛП и покажем, что релаксация полученной задачи даёт нижнюю оценку не хуже, чем LB_4. Для j-го столбца матрицы (c_{ij}) и соответствующей (3.1) перестановки $(i_1^j, ..., i_m^j)$ вычислим величины

$$\Delta_l^j = \min\{0; c_{i_{l+1}j} - c_{i_lj}\}, l = 1, \ldots, m-1.$$

Пусть $L_j = \{l \in \{1, ..., m-1\} \mid \Delta_l^j < 0\}$. Заметим, что при заданном $j \in J$ по номеру $l \in L_j$ однозначно восстанавливается номер $i_l \in I$. Заменим j-й столбец матрицы (a_{ij}) на $|L_j|$ столбцов вида

$$\overline{a}_{il} = \begin{cases} 0, & \text{если } \ i \in T_{i_lj} \\ -\Delta_l^j, & \text{если } \ i \notin T_{i_lj}. \end{cases} \quad i \in I, \ l \in L_j.$$

Тогда

$$a_{ij} = \sum_{l \in L_j} (\overline{a}_{il} + \Delta_l^j), i \in I, j \in J$$

и задача о паре матриц может быть переписана в виде: найти

$$\min_{y_i \in \{0,1\}, \sum_{i \in I} y_i = p} \left\{ \sum_{j \in J} \sum_{l \in L_j} \max_{i | y_i = 1} \overline{a}_{il} + \sum_{j \in J} \min_{i | y_i = 1} b_{ij} \right\} + \sum_{j \in J} \sum_{l \in L_j} \Delta_l^j. \qquad (3.21)$$

Введём дополнительные переменные $v_l^j \in \{0, 1\}$, $l \in L_j$ и $j \in J$, и представим эту задачу следующим образом: найти

$$\min \left\{ \sum_{j \in J} \sum_{l \in L_j} -\Delta_l^j v_l^j + \sum_{i \in I} \sum_{j \in J} b_{ij} x_{ij} \right\} + \sum_{j \in J} \sum_{l \in L_j} \Delta_l^j \qquad (3.22)$$

при ограничениях:

$$y_i \le x_{ij} + \sum_{k \in S_{ij}} x_{kj}, \ j \in J, \ i \in I, \qquad (3.23)$$

$$\sum_{i \in I} x_{ij} = 1, \ j \in J, \qquad (3.24)$$

$$\sum_{i \in I} y_i = p, \qquad (3.25)$$

$$0 \le x_{ij} \le y_i, \ j \in J, \ i \in I, \qquad (3.26)$$

$$v_l^j \geq \sum_{k \notin T_{i_l j}} x_{kj'},\ l \in L_j,\ j' \in J, j \in J, \tag{3.27}$$

$$v_l^j, y_i, x_{ij} \in \{0,1\},\ l \in L_j, j \in J,\ i \in I. \tag{3.28}$$

Обозначим через LB_6 оптимальное значение линейной релаксации этой задачи.

Теорема 16 $LB_6 \geq LB_4$.

Доказательство. Покажем, что задача (3.2), (3.10), (3.3)–(3.7) эквивалентна задаче (3.22)–(3.26), (3.28) с дополнительным ограничением

$$v_l^j = \sum_{k \notin T_{i_l j}} x_{kj},\ l \in L_j,\ j \in J. \tag{3.29}$$

Так как $c_{ij} = a_{ij} + b_{ij}$, где $i \in I$ и $j \in J$, то

$$\sum_{i \in I} c_{ij} x_{ij} = \sum_{l \in L_j} -\Delta_l^j v_l^j + \sum_{l \in L_j} \Delta_l^j + \sum_{i \in I} b_{ij} x_{ij},$$

для всех $j \in J$ и допустимого решения v_l^j, x_{ij} задачи (3.22)–(3.26), (3.28), (3.29). Действительно,

$$\sum_{l \in L_j} -\Delta_l^j v_l^j + \sum_{l \in L_j} \Delta_l^j = \sum_{l \in L_j} \Delta_l^j (1 - v_l^j) = \sum_{l \in L_j} \Delta_l^j \sum_{k \in T_{i_l j}} x_{kj} =$$

$$\sum_{k=2}^{m} x_{i_k j} \sum_{l=1}^{k-1} \Delta_l^j = \sum_{k=2}^{m} a_{i_k j} x_{i_k j} = \sum_{i \in I} a_{ij} x_{ij}.$$

Заметим, что это равенство верно как для целочисленных значений переменных, так и для дробных. Следовательно, задача (3.2), (3.10), (3.3)–(3.7) эквивалентна задаче (3.22)–(3.26), (3.28), (3.29). Тогда значение LB_4 равно оптимальному значению линейной релаксации задачи (3.22)–(3.26), (3.29), причём если ограничения (3.29) заменить на ограничения

$$v_l^j \geq \sum_{k \notin T_{i_l j}} x_{kj},\ l \in L_j,\ j \in J, \tag{3.30}$$

то оптимальное значение не изменится, так как рассматривается задача на минимум, коэффициенты в целевой функции (3.22) при v_l^j положительны ($-\Delta_l^j \geq 0$). Следовательно, значения v_l^j однозначно определяются по переменным x_{ij}. Но ограничения (3.30) составляют часть ограничений (3.27), следовательно, $LB_4 \leq LB_6$. $\square$.

Теорема 17 *Для любого $N > 0$ существуют исходные данные задачи МПК такие, что $LB_6/LB_4 \geq N$.*

Доказательство. Пусть $k = N$ и исходные данные те же, что и в доказательстве теоремы 15. Тогда решение

$$y_i = \begin{cases} 1, & \text{если} \quad i < p; \\ 0, & \text{если} \quad i = p; \\ 1/k, & \text{если} \quad i > p; \end{cases}$$

$$x_{ij} = \begin{cases} 1 - 1/k, & \text{если} \quad i = p - 1, & j = 1; \\ 1/k, & \text{если} \quad i = p + k, & j = 1; \\ 0, & \text{если} \quad i \neq p - 1; i \neq p + k, & j = 1; \\ 0, & \text{если} \quad i \leq p, & j = 2; \\ 1/k, & \text{если} \quad i > p, & j = 2; \end{cases}$$

является допустимым для задачи (3.2), (3.3)–(3.6), (3.10). Следовательно, $LB_4 \leq 1/k$. Двойственная задача к задаче (3.22)– (3.28) имеет вид: найти

$$\max \left\{ \sum_{j \in J} u_j + pr \right\} + \sum_{j \in J} \sum_{l \in L_j} \Delta_l^j$$

при ограничениях:

$$\sum_{j' \in J} t_{lj'} \leq -\Delta_l^j, \quad l \in L_j, j \in J,$$

$$\sum_{j \in J} (w_{ij} - \alpha_{ij}) + r \leq 0, \quad i \in I,$$

$$\alpha_{ij} + \sum_{k \in T_{ij}} \alpha_{kj} + u_j - w_{ij} - \sum_{j' \in J} \sum_{l \in L_{j'}} (t_{lj'} | l \notin T_{ij'}) \leq b_{ij}, \quad i \in I, j \in J,$$

$$w_{ij} \geq 0, \alpha_{ij} \geq 0, t_{lj'} \geq 0, \quad l \in L_j, j \in J, j' \in J, i \in I.$$

Для приведённых исходных данных имеем: $L_j = \{1\}, \Delta_l^j = -k$ при $j = 1, L_j = \varnothing$ при $j = 2$ и

$$\bar{a}_{ij} = \begin{cases} 0, & \text{если} \quad i < p, \ \ j = 1; \\ k, & \text{если} \quad i \geq p \quad j = 1; \end{cases} \qquad b_{ij} = \begin{cases} k, & \text{если} \quad i \leq p, \ \ j = 1; \\ 1, & \text{если} \quad i > p, \ \ j = 1; \\ 1, & \text{если} \quad i < p, \ \ j = 2; \\ 0, & \text{если} \quad i \geq p, \ \ j = 2. \end{cases}$$

Решение

$$w_{ij} = \alpha_{ij} = r = 0, u = (k, 1), t_{lj'} = \begin{cases} k - 1, & \text{если} \quad l = 1 \text{ и } j' = 1; \\ 1, & \text{если} \quad l = 1 \text{ и } j' = 2; \end{cases}$$

является допустимым в двойственной задаче. Следовательно, $LB_6 \geq k + 1 - k = 1$. □

Сравнение нижних оценок

В таблице 8 приводятся результаты расчетов для тридцати тестовых примеров из электронной библиотеки. Наряду с указанными нижними оценками линейного программирования приведена еще оценка LB_7, полученная точным решением задачи о p-медиане без учета предпочтений клиентов. Очевидно, что это действительно нижняя оценка, т. к. учет предпочтений клиентов может только увеличить значение оптимума. Для сравнения приводятся еще верхние оценки, полученные генетическим алгоритмом (GA)(см. раздел 3.4.2), оптимальные значения целевой функции (Opt), полученные методом ветвей и отсечений коммерческим пакетом XPress, разрыв (Gap) между оптимумом и наилучшими нижними оценками в процентах.

Заметим, что решение, найденное генетическим алгоритмом за несколько секунд, на всех тестовых примерах совпало с оптимальным решением задачи, полученным с помощью коммерческого пакета за несколько часов работы. Разрыв между оптимумом и нижней оценкой достаточно велик, что оставляет поле для дальнейших исследований. Возможные пути усиления нижней оценки связаны, в частности, с введением так называемых *правильных* неравенств. В [14, 15] такие неравенства получены с помощью известных результатов для задачи об упаковке множеств.

Таб. 8 *Сравнение верхних и нижних оценок*

Код задачи	LB_6	LB_4	LB_7	GA	Opt	Gap(%)
333	118,6	113,1	147	172	172	17,01
433	115,9	110,5	145	156	156	7,59
533	123,6	119,1	177	188	188	6,21
633	118,5	112,2	144	165	165	14,60
733	112,0	105,7	137	159	159	16,06
833	122,5	118,5	144	170	170	18,06
933	110,4	104,8	130	160	160	23,08
1033	107,6	104,4	138	159	159	15,22
1133	114,6	109,1	147	163	163	10,88
1233	112,4	108,2	142	163	163	14,79
1333	114,5	108,6	140	168	168	20
1433	113,6	108,2	152	172	172	13,16
1533	105,5	101,4	133	152	152	14,29
1633	110,9	105,5	141	156	156	10,64
1733	104,8	99,6	134	152	152	13,43
1833	110,6	103,9	139	154	154	10,79
1933	114,3	108,6	137	158	158	15,33
2033	112,7	108,0	140	161	161	15,00
2133	120,3	113,2	138	166	166	20,29
2233	108,0	101,5	121	154	154	27,27
2333	113,1	105,3	133	155	155	16,54
2433	115,2	109,8	139	155	155	11,51
2533	105,4	99,0	131	147	147	12,21
2633	110,4	104,3	132	156	156	18,18
2733	113,3	109,3	139	159	159	14,39
2833	107,8	101,8	137	161	161	17,52
2933	104,3	97,2	124	152	152	22,58
3033	112,2	108,1	137	157	157	14,60
3133	103,8	99,4	141	155	155	9,93
3233	108,2	102,7	129	155	155	20,16

3.5 Гибридные методы

Идея одновременного использования разных подходов к решению NP-трудных задач уходит корнями в прошлое столетие. Различные эвристические алгоритмы, по разному решая комбинаторные задачи, неизбежно будут иметь свои сильные и слабые стороны. В гибридных методах разработчики стараются воспользоваться сильными сторонами каждого метода и сгладить их недостатки. Это желание естественно. Вопрос только в том, как именно конструировать гибридные методы, чтобы достичь наибольшей эффективности.

Последние 15 лет интерес к гибридным методам постоянно возрастает. Практически на всех конференциях по исследованию операций этому направлению уделяется пристальное внимание. Появилась даже специализированная конференция по гибридизации метаэвристик. Во многих успешных приложениях фактически используются идеи сочетания разных метаэвристик, т.е. гибридных методов. Среди способов гибридизации можно выделить следующие:

– использование компонент одной метаэвристики в рамках другой;

– параллельное использование разных метаэвристик с обменом информации между ними (кооперативный поиск);

– интеграция метаэвристик с систематическим (полным или частичным) поиском.

Подробную классификацию гибридных метаэвристик можно найти в [202].

В настоящем разделе представлен один из возможных вариантов гибридной схемы, сочетающий в себе элементы вероятностного поиска с запретами и генетических алгоритмов. Этот подход будет применен к конкурентной задаче о p-медиане для нахождения приближенного решения с апостериорной оценкой точности. Интересно, что в предложенном подходе метаэвристики используются для получения как нижних, так и верхних оценок оптимума в рассматриваемой задаче на максимум. Исходная задача двухуровневого программирования переписывается в виде задачи ЦЛП с экспоненциальным числом переменных и ограничений. С помощью метаэвристик выбирается небольшое семейство переменных и ограничений, что приводит к оценочной задаче. Ее оптимальное решение дает верхнюю оценку оптимума в исходной задаче. Более того, эту схему можно расширить и получить точный метод, на каждой итерации которого требуется решение некоторой оценочной задачи. Такая схема

позволяет точно решать задачи малой размерности при $n = m \leq 30$. При бóльшей размерности, когда $n = m = 100$, $p = 10$, наблюдаемая оценка относительной погрешности составляет не более $10\% - 15\%$.

3.5.1 Гибридный генетический локальный поиск

Напомним, что конкурентная задача о p-медиане из раздела 1.4 может быть представлена следующим образом: найти

$$\max_{x_i} \sum_{j \in J} w_j u_j^*(x_i, y_i^*)$$

при условиях

$$\sum_{i \in I} x_i = p,$$

$$x_i \in \{0, 1\}, \quad i \in I,$$

где $u_j^*(x, y^*), y^*$ — оптимальное решение задачи Конкурента: найти

$$\max_{y_i, u_j} \sum_{j \in J} w_j (1 - u_j)$$

при условиях

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

$$1 - u_j \leq \sum_{i \in I_j(x)} y_i, \quad j \in J,$$

$$y_i + x_i \leq 1, \quad i \in I,$$

$$y_i, u_j \in \{0, 1\}, \quad i \in I, j \in J.$$

Переменные этой задачи имеют следующий смысл:

$$x_i = \begin{cases} 1, & \text{если Лидер открывает } i\text{-е предприятие,} \\ 0 & \text{в противном случае,} \end{cases}$$

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

$$u_j = \begin{cases} 1, & \text{если } j\text{-й клиент обслуживается Лидером,} \\ 0 & \text{если } j\text{-й клиент обслуживается Конкурентом.} \end{cases}$$

Пусть x — допустимое решение задачи и y, u —оптимальное решение задачи Конкурента для заданного x. Тогда значение $\sum_{j \in J} w_j u_j$ является нижней оценкой оптимума в исходной задаче двухуровневого программирования. Так как каждый клиент обязательно обслуживается либо Лидером, либо Конкурентом, то можно считать, что нижняя оценка полностью определяется вектором x. Оставшаяся часть этого раздела посвящена именно этой проблеме, выбору вектора x. Нахождение оптимальных значений y и u для x требует точного решения задачи Конкурента. Известно [139], что она является NP-трудной. Для вычисления y, u используется коммерческий пакет CPLEX. Если разрыв целочисленности оказывается небольшим, как это имеет место для евклидовых задач размещения на плоскости, то такой подход позволяет быстро найти точное решение задачи Конкурента.

Первая и, возможно, простейшая стратегия нахождения вектора x состоит в игнорировании Конкурента. Лидер открывает свои предприятия, минимизируя суммарное взвешенное расстояние от клиентов до предприятий. Он хочет обслужить всех клиентов и решает классическую задачу о p-медиане [163]. Полученный вектор x порождает нижнюю оценку. Обозначим ее через $\mathrm{LB}(p)$. Эта стратегия является разумной в том смысле, что Лидер стремится расположить свои предприятия как можно ближе к клиентам. Потом, когда на рынок выйдет Конкурент, ему будет трудно "отрезать" Лидера от клиентов. Как мы увидим позже, эта стратегия не является наилучшей из возможных. Лидер теряет, как правило, больше половины рынка при $p = r$, но это решение часто оказывается достаточно хорошим для евклидовых примеров.

Вторая стратегия является более продвинутой в плане учета действий Конкурента. Лидер знает, что вслед за ним на рынок придет Конкурент и откроет свои r предприятий. Тем самым будет открыто $p + r$ предприятий. При централизованном планировании это потребует бы решения задачи о $(p + r)$-медиане. Для каждого предприятия в оптимальном решении можно вычислить зоны обслуживания и найти получаемый доход. Вторая стратегия состоит в решении задачи о $(p + r)$-медиане и выборе для Лидера p наиболее доходных предприятий. Получаемую таким способом нижнюю оценку обозначим через $\mathrm{LB}(p+r)$. Предложенная стратегия кажется вполне разумной, но ее результат для Лидера очень слабый. Конкурент стремится к собственной выгоде, он старается отсечь Лидера от клиентов. Его решение никак не связано с решением задачи

о $(p + r)$-медиане. Результаты расчетов показывают, что эта стратегия сильно проигрывает предыдущей.

Третья стратегия состоит в применении метаэвристик к решению исходной задачи двухуровневого программирования. Рассмотрим следующий гибридный алгоритм локального поиска, сочетающий в себе идеи генетических алгоритмов и поиска с запретами. Ниже представлена общая схема такого подхода.

Гибридный генетический локальный поиск

1. Создать начальную популяцию решений Лидера.
2. Пока не выполнен критерий остановки, делать следующее.
   2.1. Выбрать два решения x^1, x^2 из популяции.
   2.2. Построить решение x' по решениям x^1, x^2.
   2.3. Модифицировать x'.
   2.4. Применить алгоритм PTS, используя x' как стартовое решение.
   2.5. Наилучшее решение, полученное PTS, добавить в популяцию и наихудшее удалить.
3. Предъявить наилучшее найденное решение.

Выполнение шагов 2.1. – 2.5. будем называть итерацией. Выполнение заданного числа итераций будет использоваться в качестве критерия остановки алгоритма.

Начальная популяция

Как уже отмечалось в предыдущем разделе, выбор хорошей начальной популяции может заметно сократить время получения точного решения задачи и приводит к решениям с малой погрешностью даже при небольшом числе итераций. Как и в случае задачи МПК, в качестве начальной популяции целесообразно взять множество локальных оптимумов относительно полиномиально проверяемых окрестностей, например, Swap-окрестности. Она хорошо зарекомендовала себя при решении классической задачи о p-медиане [171]. Эта окрестность содержит $p(n - p)$ элементов. Для каждого из них требуется вычисление целевой функции, что приводит к точному решению задачи Конкурента. Таким образом, каждый шаг локального улучшения требует решения серии NP-трудных задач. Чтобы сократить трудоемкость процедуры локального улучшения, заменим условие целочисленности переменных в задаче Конкурента на условие их принадлежности отрезку [0, 1]. Получаем задачу линейно-

го программирования. Теперь просмотр окрестности и выбор наилучшего элемента в ней становится полиномиальной процедурой. Если разрыв целочисленности невелик, то такая модификация хотя и не гарантирует получения локального минимума, но приводит к хорошей начальной популяции. В целях диверсификации процедура локального улучшения применяется к стартовым решениям, выбранным случайным образом. Более того, при добавлении очередного решения в популяцию вычисляется расстояние Хэмминга от него до каждого решения из популяции. Новое решение добавляется в популяцию, если минимальное из расстояний оказывается не меньше заданного порога $\lceil 0,6p \rceil$.

Основные операторы и параметры

Остановимся на основных шагах 2.1.–2.5., выполняемых на каждой итерации алгоритма. Выбор родительской пары x^1, x^2 на шаге 2.1. производится согласно известному правилу турнирной селекции [193]. Из популяции случайным образом с равномерным распределением выбирается k решений и лучшее из них по целевой функции объявляется родителем. В численных экспериментах полагалось $k = 5$ и проверялось условие $x^1 \neq x^2$. Операция скрещивания дает равные права родительским решениям (uniform crossover). Новое решение x' получает общие компоненты родительских решений x^1, x^2. Оставшаяся часть все еще неоткрытых предприятий выбирается из родительских решений случайным образом с равномерным распределением.

В целях диверсификации поиска на шаге 2.3. применяется процедура случайных модификаций (мутации). Так как решение x' после этой процедуры должно остаться допустимым, то в качестве мутаций используется малое число шагов в случайном направлении по окрестности Swap.

На последнем шаге 2.5. происходит обновление популяции. Новое решение, полученное вероятностным алгоритмом поиска с запретами, добавляется в популяцию. Наихудшее по целевой функции решение удаляется из популяции. При добавлении нового решения производится следующая проверка. Если по расстоянию Хэмминга новое решение оказывается ближе чем на $\lceil 0,6p \rceil$ от одного из элементов популяции, то это решение выбрасывается как неперспективное, и итерация на этом считается завершенной.

Поиск с запретами

В алгоритмах генетического локального поиска [173, 153] на шаге 2.4. используются, как правило, стандартные процедуры локального улучшения. В результате поиск концентрируется на множестве локальных оптимумов, что положительно сказывается на результатах расчетов. Однако локальных оптимумов в задаче может оказаться очень много (экспоненциальное число от размерности задачи), и они могут сильно отличаться друг от друга по значению целевой функции. Например, для задачи о p-медиане эта разница в общем случае не может быть ограничена константой [40]. Поэтому останавливать поиск в первом найденном локальном оптимуме может оказаться нецелесообразным. Кроме того, в наших экспериментальных исследованиях при $w_j = 1, j \in J$, целевая функция может принимать только целые значения от 1 до $|J|$. В результате появляется масса разных локальных оптимумов с одинаковыми значениями целевой функции, которые образуют своего рода *плато*. Попадая на любое из них такой алгоритм не может найти путь для улучшения целевой функции. Для преодоления этой трудности требуется более мощные средства, чем стандартные алгоритмы локального улучшения. Именно по этой причине на шаге 2.4. применяется другая метаэвристика из класса траекторных методов, вероятностный поиск с запретами, который позволяет "выбираться" из локальных оптимумов и систематически находить решения с все меньшей и меньшей погрешностью.

Основная проблема при реализации такого подхода состоит в просмотре окрестности и выборе в ней наилучшего элемента. Это самая трудоемкая часть. Поэтому приходится ограничиваться окрестностями малой мощности, например, окрестностью Swap, и использовать ее малую часть, выбранную тем или иным способом. В предыдущем разделе исследовался один из таких способов, когда эта часть выбиралась случайным образом с равномерным распределением. Такой подход позволяет резко сократить трудоемкость, ослабить зависимость от длины списка запретов и вносить элемент диверсификации, столь необходимый при наличии плато.

Наряду с рандомизацией окрестности для сокращения трудоемкости одного шага алгоритма PTS используется также переход к линейной релаксации при вычислении целевой функции Лидера. Замена условий целочисленности переменных на условие их принадлежности отрезку [0, 1] слегка завышает доход Конкурента. В результате получается оценка сни-

зу на доход Лидера. Элемент окрестности с наибольшей нижней оценкой считается наилучшим элементом в окрестности. Такой подход, как уже отмечалось, делает процедуру просмотра окрестности полиномиальной. Ниже приводится общая схема алгоритма PTS.

Алгоритм PTS

1. Построить начальное решение x, положить $Tabu = \emptyset$, выбрать коэффициент рандомизации окрестности $q > 0$.
2. Пока не выполнен критерий остановки, делать следующее.
   2.1. Построить окрестность $Swap_q(x)$.
   2.2. Если $Swap_q(x) \setminus Tabu \neq \emptyset$, то найти наилучший элемент x' в множестве $Swap_q(x) \setminus Tabu$, иначе положить $x' = x$.
   2.3. Перейти от x к x'.
   2.4. Перестроить список запретов $Tabu$.
3. Предъявить наилучшее найденное решение.

Множество $Tabu$ для текущего решения x содержит часть элементов окрестности $Swap(x)$. Пары предприятий, вводящиеся и выводящиеся из решений на нескольких последних шагах алгоритма, запоминаются в ходе поиска. На каждой итерации они порождают множество $Tabu$, которое удаляется из окрестности. Число таких пар (длина списка запретов) выбирается таким образом, чтобы предотвратить зацикливание. Кроме того, в ходе работы алгоритма порождается семейство "хороших" решений Конкурента. Как будет показано ниже, любое такое семейство позволяет получить верхнюю оценку на оптимум исходной задачи.

3.5.2 Верхние оценки

Покажем, как с помощью ЦЛП можно получить семейство верхних оценок для оптимального значения целевой функции Лидера и, соответственно, нижних оценок для оптимального значения целевой функции Конкурента. Рассмотрим два способа получения таких оценок, причем второй способ будет давать также возможность найти точное решение задачи.

Введение дополнительных ограничений

Общая идея построения таких оценок выглядит следующим образом. Добавим в систему ограничений задачи Конкурента некоторое дополни-

тельное ограничение. При этом оптимум в задаче Конкурента не увеличится, а оптимум в задаче Лидера может только возрасти. Если введение дополнительного ограничения позволяет свести исходную задачу двухуровневого программирования к ЦЛП, то ее оптимальное решение дает нужную оценку. Проблема состоит в том, чтобы найти подходящее ограничение, обладающее требуемым свойством.

Предположим, что Конкурент при размещении предприятий действует по следующему правилу. Он упорядочивает возможные места размещения предприятий согласно некоторому критерию. Это упорядочение производится перед решением задачи и доводится до сведения Лидера. Как только Лидер объявляет свое решение, Конкурент размещает свои предприятия согласно выбранному порядку в тех местах, которые не занял Лидер. Такая стратегия не гарантирует нахождения оптимального решения Конкурента и, следовательно, приводит к нижней оценке оптимума для Конкурента. Эта нижняя оценка зависит от выбранного упорядочения. Меняя упорядочение, получаем разные нижние оценки. Таким образом, получаем семейство из $m!$ нижних оценок для Конкурента или верхних оценок на оптимум Лидера.

Представим сведение задачи с указанным ограничением на поведение Конкурента к ЦЛП. Без ограничения общности будем считать, что возможные места размещения предприятий уже упорядочены в соответствии с предпочтениями Конкурента. Первое предприятие $i = 1$, является наиболее желательным для него, последнее предприятие $i = m$, — наименее желательным. Тогда при выбранных значениях x_i, $i \in I$, поведение Конкурента однозначно определяется следующей системой ограничений:

$$x_i + y_i \le 1, \quad i \in I,$$

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

$$x_i + y_i \ge y_k, \quad i, k \in I, k > i.$$

Последнее неравенство запрещает Конкуренту открывать предприятие в пункте k, если хотя бы одно из предприятий с меньшим номером не было открыто Лидером или Конкурентом. Введем дополнительные переменные:

$$z_{jk} = \begin{cases} 1, & \text{если клиент } j \text{ обслуживается из предприятия } k \\ 0 & \text{в противном случае.} \end{cases}$$

Переменные z_{jk} однозначно определяются по значениям x_i и y_i из следующей системы ограничений:

$$\sum_{k \in I} z_{jk} = 1, \quad j \in J,$$

$$z_{jk} \leq x_k + y_k, \quad j \in J, k \in I,$$

$$z_{jk} + \sum_{l \in S_{kj}} z_{jl} \geq x_k + y_k, \quad k \in I, j \in J.$$

Первое равенство требует обслужить каждого клиента из предприятий Лидера или Конкурента. Второе ограничение разрешает обслуживание только из открытых предприятий. Последнее неравенство устанавливает приоритет в выборе предприятий при обслуживании клиентов. Согласно этому ограничению наиболее предпочтительное из открытых предприятий Лидера или Конкурента будет выбрано для обслуживания данного клиента. Кроме того, это ограничение доминирует первое ограничение в предыдущей системе, так как $z_{jk} + \sum_{l \in S_{kj}} z_{jl}$ всегда не больше единицы.

При заданных значениях z_{jk}, x_i, y_i переменные u_j однозначно определяются следующей системой ограничений:

$$u_j \geq z_{jk} - y_k, \quad j \in J, k \in I,$$

$$1 - u_j \geq z_{jk} - x_k, \quad j \in J, k \in I.$$

Первое неравенство требует включить j-го клиента в область обслуживания Лидера, если $z_{jk} = 1$ и $y_k = 0$. Если же $z_{jk} = 1$ и $y_k = 1$, то второе неравенство требует включить этого клиента в область обслуживания Конкурента.

Таким образом, выбрав значения x_i, $i \in I$, можно однозначно определить значения y_i, $i \in I$, z_{jk}, $j \in J, k \in I$ и u_j, $j \in J$. Оптимальные значения всех переменных определяются следующей задачей: найти

$$\max \sum_{j \in J} w_j u_j$$

при условиях

$$\sum_{i \in I} x_i = p,$$

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

$$x_i + y_i \geq y_k, \quad i, k \in I, k > i,$$

$$\sum_{k\in I} z_{jk} = 1, \quad j \in J,$$

$$z_{jk} \le x_k + y_k, \quad j \in J, k \in I,$$

$$z_{jk} + \sum_{l\in S_{kj}} z_{jl} \ge x_k + y_k, \quad k \in I, j \in J,$$

$$u_j \ge z_{jk} - y_k, \quad j \in J, k \in I,$$

$$1 - u_j \ge z_{jk} - x_k, \quad j \in J, k \in I,$$

$$x_i, y_i, z_{jk}, u_j \in \{0, 1\}, \quad k, i \in I, j \in J.$$

Пусть $(x_i^*, y_i^*, z_{jk}^*, u_j^*)$ — оптимальное решение сформулированной задачи. Оно дает верхнюю оценку UB_0 на оптимум Лидера. Линейная релаксация данной задачи, полная или частичная, также дает верхнюю оценку. Для получения нижней оценки достаточно решить задачу Конкурента при заданных значениях x_i^*. В этом случае получаем приближенное решение исходной задачи двухуровневого программирования с апостериорной оценкой точности.

Сужение области поиска

Представим другой подход к получению верхней оценки, который принципиально отличается от предшествующего. Исходную задачу двухуровневого программирования эквивалентным образом перепишем в терминах линейного частично-целочисленного программирования с экспоненциальным числом переменных и ограничений. Часть этих переменных и ограничений удалим из задачи, что приведет к требуемой верхней оценке. Обозначим через $\mathcal{F}$ непустое семейство допустимых решений задачи Конкурента. Будем предполагать, что Кокурент выбирает свое решение, ограничиваясь только семейством $\mathcal{F}$. Для $y \in \mathcal{F}$ положим

$$I_j(y) = \{i \in I \mid g_{ij} \le \min_{l\in I}(g_{lj} \mid y_l = 1)\}, \quad j \in J.$$

Это множество показывает места размещения производства, которые позволяют Лидеру "удержать" клиента j, если Конкурент использует решение y. Введем новые переменные:

$$u_{jy} = \begin{cases} 1, & \text{если } j\text{-й клиент обслуживается Лидером, а Конкурент} \\ & \text{использует решение } y, \\ 0, & \text{если } j\text{-й клиент обслуживается Конкурентом,} \\ & \text{и Конкурент использует решение } y, \end{cases}$$

$W \geq 0$ — суммарный доход Лидера.

Для данного семейства $\mathcal{F}$ задача Лидера может быть представлена следующим образом: найти

$$\max W$$

при условиях:

$$\sum_{j \in J} w_j u_{jy} \geq W, \quad y \in \mathcal{F},$$

$$u_{jy} \leq \sum_{i \in I_j(y)} x_i, \quad j \in J, y \in \mathcal{F},$$

$$\sum_{i \in I} x_i = p,$$

$$x_i, u_{jy} \in \{0, 1\}, \quad i \in I, j \in J, y \in \mathcal{F}.$$

Целевая функция $W(\mathcal{F})$ задачи определяет суммарный доход Лидера. Первое ограничение гарантирует, что Конкурент выберет наилучший ответ из семейства $\mathcal{F}$ на любой ход Лидера. Второе ограничение определяет раздел рынка между Лидером и Конкурентом. Если семейство $\mathcal{F}$ содержит все возможные решения Конкурента, то оптимальное решение данной задачи дает оптимальное решение исходной задачи двухуровневого программирования.

Заметим, что замена булевых переменных u_{jy} на непрерывные переменные $0 \leq u_{jy} \leq 1$ не меняет оптимального решения задачи. Действительно, если в оптимальном решении задачи с непрерывными переменными какая-то компонента u_{jy} оказалась дробной, то увеличение ее до единицы не нарушает второго ограничения и способствует увеличению максимальной величины W. Таким образом, в задаче останется только одна группа булевых переменных. Остальные переменные можно считать непрерывными. Такие задачи линейного частично–целочисленного программирования можно решать точно с помощью коммерческого программного обеспечения, если семейство $\mathcal{F}$ достаточно мало. Однако, каково бы ни было семейство $\mathcal{F}$, оно всегда приводит к верхней оценке для оптимума в исходной задаче двухуровневого программирования. Обозначим эту верхнюю оценку $UB_1(\mathcal{F})$.

Предположим, что семейство $\mathcal{F}$ содержит все возможные решения Конкурента и для этого семейства удалось точно решить сформулированную задачу. Интуитивно ясно, что многие из элементов семейства $\mathcal{F}$

будут порождать несущественные ограничения, и семейство $\mathcal{F}$ можно сузить без изменения оптимума задачи. Вычисление такого подсемейства представляется сложной проблемой. Одним из путей ее решения, а заодно и точным решением исходной задачи может, служить следующий итеративный процесс.

Пусть $\mathcal{F}$ — некоторое семейство решений Конкурента и $x(\mathcal{F})$ — оптимальное решение задачи Лидера при данном семействе. Тогда величина $W(\mathcal{F})$ задает верхнюю оценку оптимума исходной задачи двухуровневого программирования, а оптимальное решение $y(\mathcal{F})$ задачи Конкурента для решения $x(\mathcal{F})$ определяет нижнюю оценку оптимума. Обозначим эту нижнюю оценку через $LB(\mathcal{F})$. Теперь поиск оптимума и требуемого подсемейства можно получить с помощью следующего метода пополнения семейства $\mathcal{F}$.

Точный метод

1. Выбрать начальное семейство $\mathcal{F}$.
2. Найти $W(\mathcal{F})$ и $x(\mathcal{F})$.
3. Решить задачу Конкурента и вычислить $LB(\mathcal{F})$.
4. Если $W(\mathcal{F}) = LB(\mathcal{F})$, то STOP.
5. Добавить $y(\mathcal{F})$ в семейство $\mathcal{F}$ и вернуться на шаг 2.

Убедимся в том, что метод является конечным и точным. Из конечности множества допустимых решений задачи Конкурента следует конечность самого метода. Единственная возможность не получить точное решение состоит в зацикливании метода, когда добавление на шаге 5 решения $y(\mathcal{F})$ к семейству $\mathcal{F}$ не меняет само семейство, решение $y(\mathcal{F})$ уже принадлежало семейству. Но тогда $W(\mathcal{F})$ не может быть больше $LB(\mathcal{F})$, как следует из первой группы ограничений задачи для поиска $W(\mathcal{F})$. Значит, $LB(\mathcal{F}) = W(\mathcal{F})$ и метод останавливается, получив точное решение $x(\mathcal{F})$ исходной задачи двухуровневого программирования.

К сожалению, получить нетривиальные оценки на число итераций такого метода вряд ли удастся. Кроме того, на каждой итерации приходится решать задачу поиска $W(\mathcal{F})$ для данного семейства $\mathcal{F}$, что также может оказаться весьма непростым делом. Тем не менее, даже несколько шагов этого метода при удачном выборе начального семейства могут приводить к хорошей верхней оценке оптимума. Один из способов выбора начального семейства предложен в предыдущем разделе.

3.5.3 Тестовые расчеты

Для сравнения предложенных верхних оценок был проведен численный эксперимент на тестовых примерах из электронной библиотеки «Дискретные Задачи Размещения» (http://math.nsc.ru/AP/benchmarks). Для всех примеров $n = m = 100$, $w_j = 1$, $j \in J$, $p = r = 10$. Элементы матрицы g_{ij} определялись как евклидовы расстояния между точками i, j на двухмерной плоскости. Точки выбирались из квадрата 7000×7000 случайным образом с равномерным распределением. Все эксперименты проводились на персональном компьютере PC Pentium Intel Core 2, 1.87 GHr, RAM 2Gb. В Таблице 9 представлены результаты расчетов нижних и верхних оценок. В первой колонке приводится код примера. Во второй, третьей и четвертой колонках представлены нижние оценки. В двух последних колонках — полученные верхние оценки.

Табл. 9 *Сравнение нижних и верхних оценок*

Code	$LB(p)$	$LB(p+r)$	HMA	UB_0	UB_1
111	41	31	50	74	55
211	41	36	49	70	55
311	46	41	47	73	55
411	41	39	48	72	55
511	48	40	48	70	55
611	42	37	47	67	55
711	49	37	51	73	55
811	42	37	48	74	55
911	47	35	49	68	55
1011	46	33	49	70	55

Сравнение нижних оценок показывает доминирование оценки HMA, полученной гибридным генетическим локальным поиском, над другими оценками. Алгоритм находит данные значения за несколько десятков итераций. Дальнейшие многочасовые расчеты не приводят к улучшению. Скорее всего, полученные решения являются оптимальными. Алгоритм многократно получает их при разных наборах открываемых предприятий. Для всех примеров полученные значения оценки HMA соответствуют сотням решений Лидера. Однако доказать их оптимальность пока не удается. Оценка $LB(p)$ мало проигрывает оценке HMA и значительно

превосходит оценку $LB(p+r)$. Так как для ее получения требуется только один раз решить задачу о медиане, то такой вариант можно считать хорошим способом для получения стартового решения для алгоритмов поиска с запретами. Интересно отметить, что в ходе численных экспериментов получаемые значения оценки HMA оказались нечувствительными к замене условий целочисленности переменных в задаче Конкурента на условия их принадлежности отрезку [0,1]. Разрыв целочисленности при данном способе порождения исходных данных, действительно, часто оказывался нулевым, и такая замена не влияла на результаты экспериментов. Ситуация, по-видимому, может измениться, если каждый клиент будет иметь свое небольшое множество потенциальных поставщиков. Однако в этом случае задача уже не будет сводиться к поиску максимина, и понятие оптимального решения потребует дальнейшего уточнения [182].

При вычислении величины UB_0 использовалось упорядочение предприятий, полученное Лагранжевыми релаксациями. Семейство $\mathcal{F}$ для верхней оценки UB_1, порождалось методами локального поиска, как это уже отмечалось выше. Размер семейства составлял 700–800 элементов. Среднее время счета для UB_0 — 1 час, для UB_1 — 24 часа.

Вторая верхняя оценка является значительно более трудоемкой даже без пополнения начального семейства, но она гораздо точнее первой. Сравнение UB_1 с наилучшим найденным допустимым решением задачи показывает малую погрешность, около $10-15\%$. Следует заметить, что данный класс тестовых примеров является достаточно сложным для вычисления верхних оценок. Равные веса клиентов $w_j = 1, j \in J$, порождают большое число оптимальных решений, что требует большого семейства $\mathcal{F}$ в оценочной задаче для вычисления $W(\mathcal{F})$. Угадать или породить такое семейство достаточно трудно, и ошибки в определении семейства приводят к завышению верхней оценки. По-видимому, ситуация должна измениться, когда веса станут различными [91, 90]. Возможно, для такого случая удастся увидеть какие-то особенности оптимального семейства, что позволит искать его более целенаправленно. В качестве направлений для дальнейших исследований интересно также модифицировать предложенный точный метод, включив в него быстрые приближенные алгоритмы решения задач на шагах 2 и 3.

Глава 4

Вычислительная сложность локального поиска

Идеи локального поиска при решении задач комбинаторной оптимизации являются, по-видимому, наиболее естественными и наглядными. Первые шаги по их реализации относятся к концу 50-х, началу 60-х годов двадцатого столетия [54, 95, 102]. В СССР пионерами этого направления были Ю.И. Журавлев [29], предложивший алгебраическую теорию локальных алгоритмов, и Л.А. Растригин [54], исследовавший вероятностные алгоритмы локального поиска. На Западе первые исследования связаны, в основном, с задачей коммивояжера. Позднее эти идеи использовались для задач размещения, построения сетей, расписаний и др. [156, 176, 69, 56, 62]. Однако довольно быстро выяснилось, что локальный поиск не гарантирует нахождение глобального оптимума задачи. Локальные алгоритмы стали использоваться, в основном, в гибридных схемах [31], схемах декомпозиции [63, 64] и для получения приближенных решений в сложных дискретных задачах [4, 5, 55]. Предпринимались попытки модифицировать симплекс метод для нахождения точного решения специальных классов дискретных экстремальных задач [60]. Исследовались возможности построения наилучшей последовательности локальных алгоритмов [58], методы случайного поиска с локальной оптимизацией и управляемого случайного поиска [16, 46, 26, 32]. Тем не менее, отсутствие концептуального прогресса ослабило интерес к данному направлению. В последние 15–20 лет наблюдается возрождение этого подхода. Оно связано как с новыми алгоритмическими схемами, построенными на аналогиях с живой и неживой природой, так и с новыми теоретическими результатами в области локального поиска. Изменился общий взгляд на построение локальных алгоритмов. Требование монотонного улучшения целевой функции больше не является доминирующим.

Наиболее мощные вероятностные алгоритмы допускают произвольное ее ухудшение, и многие из них могут рассматриваться как способ порождения конечных неразложимых цепей Маркова на подходящем множестве состояний [34, 70]. Появление метаэвристик, таких как генетические алгоритмы, локальный поиск с запретами, имитация отжига и др. [98, 179, 189], открыло широкий простор для решения прикладных задач исследования операций.

Анализ вычислительной сложности локального поиска в последние годы интенсивно ведется в двух направлениях: эмпирическом и теоретическом. Эти направления дают существенно разные оценки возможностям локального поиска.

Эмпирические результаты. Для многих NP-трудных задач локальный поиск позволяет находить приближенные решения, близкие к глобальному оптимуму по целевой функции. Трудоемкость алгоритмов часто оказывается полиномиальной, причем степень полинома достаточно мала. Так для задачи о разбиении множества вершин графа на две равные части разработаны алгоритмы локального поиска со средней трудоемкостью $O(n^{2,4})$, где n — число вершин, которые дают всего несколько процентов погрешности [156].

Для задач коммивояжера алгоритмы локального поиска являются высоко эффективными с практической точки зрения. Один из таких алгоритмов с окрестностью Лина-Кернигана для классической задачи коммивояжера имеет погрешность в среднем около 2% и максимальная размерность решаемых задач достигает 1 000 000 городов [152, 126]. На случайно сгенерированных задачах такой колоссальной размерности итерационная процедура Джонсона позволяет находить решения с отклонением около $0,5\%$ на современных компьютерах за несколько минут. Аналогичные результаты для реальных задач размерностью до 100 000 получены в [205].

Для задач теории расписаний, размещения, покрытия, раскраски и многих других NP-трудных задач алгоритмы локального поиска показывают превосходные результаты [179, 189]. Их гибкость при изменении математической модели, простота реализации и наглядность превращают локальный поиск в мощное средство для решения NP–трудных задач.

Теоретические результаты. Исследование локального поиска с точки зрения гарантированных оценок качества показывают границы его возможностей. Построены трудные для локального поиска примеры, из ко-

торых следует, что [207, 50, 158, 40, 212]

1) минимальная точная окрестность может иметь экспоненциальную мощность;

2) число шагов для достижения локального оптимума может оказаться экспоненциальным;

3) значение локального оптимума может сколь угодно сильно отличаться от глобального оптимума;

4) минимальное расстояние между локальными оптимумами может быть сколь угодно большим.

Эти результаты подталкивают к более пристальному рассмотрению задачи нахождения локального оптимума, ее трудоемкости в худшем и среднем случаях. В данной главе приводится краткий обзор результатов в этой бурно развивающейся области дискретной оптимизации, а также основные результаты, касающиеся дискретных задач размещения. В первом разделе вводятся основные определения. Во втором разделе дается краткое введение в теорию сложности задач локального поиска. В третьем разделе устанавливается вычислительная сложность задачи поиска локального оптимума для различных задач размещения. В четвертом разделе приводится оценка в худшем случае для числа итераций алгоритмов локального спуска. В пятом разделе показана связь задачи поиска локального минимума с задачей нахождения седловых точек функции Лагранжа. В шестом разделе изучаются возможности построения приближенных локальных оптимумов и подчеркивается принципиальное отличие этой задачи от задачи поиска глобального оптимума. В последнем разделе рассматривается вопрос о погрешности локальных оптимумов и приводится семейство исходных данных, на котором расстояние от глобального минимума до ближайшего локального минимума равно диаметру допустимой области. Более того, любой путь от локального минимума в глобальный проходит через область больших значений целевой функции, что является серьезным препятствием для таких методов, как имитация отжига, поиск с запретами и др.

4.1 Задачи локального поиска

Существует несколько определений оптимизационной задачи, которые отличаются степенью детализации. Далее используется следующее определение [80].

Определение 1 *Оптимизационная задача OP определяется следующим набором объектов $< \mathcal{I}, Sol, F, goal >$, где:*

1) $\mathcal{I}$ — множество входов задачи OP;

2) Sol — функция, которая каждому входу $x \in \mathcal{I}$ сопоставляет множество допустимых решений $Sol(x)$;

3) F — функция, которая задаёт вес $F(s, x)$ допустимого решения s входа x;

4) величина $goal \in \{Min, Max\}$ уточняет, является ли задача OP задачей на максимум или минимум.

Для заданного входа x требуется найти оптимальное решение задачи OP.

В дальнейшем будем рассматривать только задачи на минимум, хотя все последующие рассуждения можно перенести и на случай задач на максимум.

Определение 2 *Задача локального поиска определяется парой $\Pi = (OP, N)$, где OP – оптимизационная задача, а N – функция, которая каждому допустимому решению s входа x сопоставляет множество $N(s, x) \subseteq Sol(x)$ соседей решения s. Множество $N(s, x)$ называют окрестностью решения s. Задача локального поиска заключается в отыскании локального минимума для заданного входа x.*

Если s — локальный минимум в задаче Π, т. е. в его окрестности нет решений с меньшим значением целевой функции, то будем говорить, что решение s является N-оптимальным. Окрестность называют точной, если любой локальный минимум является глобальным. Пусть (OP, N_1) и (OP, N_2) — две задачи локального поиска . Будем говорить, что окрестность N_2 *сильнее* окрестности N_1 (соответственно окрестность N_1 *слабее* окрестности N_2), если из N_2-оптимальности решения следует его N_1-оптимальность. Будем записывать это следующим образом: $N_1 \preceq N_2$. Введенное отношение является рефлексивным и транзитивным.

Стандартный алгоритм локального спуска начинает свою работу с некоторого начального решения s_0, выбранного случайно или с помощью какого-либо вспомогательного алгоритма. На каждом шаге локального спуска происходит переход от текущего решения к соседнему решению с меньшим значением целевой функции. Процесс происходит до тех пор, пока не будет достигнут локальный минимум.

Алгоритмы локального спуска широко применяются для решения NP-трудных задач дискретной оптимизации. Многие полиномиально разрешимые задачи могут рассматриваться как задачи, легко решаемые таким способом. При подходящем выборе полиномиальной окрестности соответствующая теорема может быть сформулирована в следующем виде: допустимое решение не является глобальным оптимумом, если и только если оно может быть улучшено некоторым локальным образом. Ниже приводится несколько примеров таких задач. Они указывают на важность локального поиска при построении оптимизационных алгоритмов и достаточно общий характер этого подхода.

1. Линейное программирование. Геометрически симплекс метод можно интерпретировать как движение по вершинам многогранника допустимой области. Вершина не является оптимальной, если и только если существует смежная с ней вершина с меньшим значением целевой функции. Алгебраически, в предположении невырожденности задачи базисное допустимое решение не является оптимальным, если и только если оно может быть улучшено локальным изменением базиса, т. е. заменой одной базисной переменной на небазисную. Получающаяся таким образом окрестность является точной и имеет полиномиальную мощность.

2. Минимальное остовное дерево. Остовное дерево не является оптимальным, если и только если локальной перестройкой, добавляя одно ребро и удаляя из образовавшегося цикла другое ребро, можно получить новое остовное дерево с меньшим суммарным весом. Операция локальной перестройки задает отношение соседства на множестве остовных деревьев. Окрестность любого дерева имеет полиномиальную мощность, а сама окрестность является точной.

3. Максимальное паросочетание. Паросочетание не является максимальным, если и только если существует увеличивающий путь. Два паросочетания называют соседними, если их симметрическая разность образует путь. Определенная таким образом окрестность является точной и имеет полиномиальную мощность. Аналогичные утверждения справедливы для взвешенных паросочетаний, совершенных паросочетаний минимального веса, задач о максимальном потоке и потоке минимальной стоимости.

На каждом шаге локального спуска окрестность задает множество возможных направлений движения. Это множество часто состоит из

нескольких элементов, и имеется определенная свобода в выборе следующего решения. Правило выбора (замещения) может оказать существенное влияние на трудоемкость алгоритма и результат его работы. Например, в задаче о максимальном потоке алгоритм Форда–Фалкерсона (который тоже можно рассматривать как вариант локального улучшения) имеет полиномиальную временную сложность при выборе кратчайшего пути для увеличения потока и экспоненциальную временную сложность без гарантии получить глобальный оптимум при произвольном выборе пути [66]. Таким образом, при разработке алгоритмов локального поиска важно не только правильно определить окрестность, но и верно задать правило замещения. Интуитивно кажется, что в окрестности надо брать элемент с наименьшим значением целевой функции. Однако разумным оказывается не только такой выбор, но и движение в «абсурдном» направлении, когда несколько шагов с ухудшением могут привести (и часто приводят) к лучшему локальному минимуму.

4.2 Класс PLS

В дальнейшем будем рассматривать только задачи локального поиска, для которых существует такой полином q, что для любого входа x длина каждого допустимого решения $s \in Sol(x)$ ограничена значением этого полинома от длины записи исходных данных, т. е. $|s| \leq q(|x|)$.

Определение 3 *Задача* П *принадлежит классу PLS, если существует три полиномиальных алгоритма A, B и C со следующими свойствами.*

1) Алгоритм A определяет, является ли слово x входом задачи. Если $x \in \mathcal{I}$, то алгоритм находит допустимое решение задачи OP.

2) Алгоритм B для любого входа $x \in \mathcal{I}$ и любого слова s определяет, является ли s допустимым решением. Если $s \in Sol(x)$, то алгоритм находит значение целевой функции за полиномиальное время.

3) Алгоритм C для любого входа $x \in \mathcal{I}$ и любого решения $s \in Sol(x)$ определяет, является ли s локальным минимумом. Если нет, то алгоритм находит соседа $s' \in N(s, x)$ с меньшим значением целевой функции.

Класс PLS не является пустым. Многие оптимизационные задачи с полиномиально проверяемыми окрестностями принадлежат этому классу. В частности, задача коммивояжера с любой окрестностью полиномиальной мощности принадлежит этому классу. Обобщенная задача о

назначениях скорее всего не принадлежит этому классу, т. к. для нее поиск допустимого решения является NP-полной задачей.

Для задач из класса PLS легко построить алгоритмы локального спуска. С помощью алгоритма A можно получить начальное допустимое решение. Алгоритм C позволяет проверить его локальную оптимальность и найти лучшее соседнее решение, если данное решение не является локальным минимумом.

В классе PLS содержатся полиномиально разрешимые задачи, т. е. задачи, в которых можно найти локальный оптимум за полиномиальное время. Многие NP–трудные задачи без весовых функций с любой полиномиально проверяемой окрестностью относятся к таковым: задачи о клике, о покрытии, о независимом множестве и др. Однако в общем случае вопрос о вычислительной сложности нахождения локального оптимума пока остается открытым. Если предположить, что NP$\neq$ co-NP, то в классе PLS нет NP-трудных задач.

Теорема 18 *[154] Если некоторая задача локального поиска из класса PLS является NP-трудной, то NP=co-NP.*

Другими словами, не существует NP-полных задач, которые за полиномиальное время сводились бы к какой-нибудь задаче локального поиска из класса PLS. Следовательно, сложность задач из этого класса меньше сложности NP-полных задач. Отметим, что гипотеза NP$\neq$ co-NP является более сильной, чем предположение о том, что P$\neq$NP, т. к. совпадение последних двух классов влечет совпадение первых двух [25].

Для задач локального поиска определяется понятие PLS-сведения.

Определение 4 *Пусть* Π_1 *и* Π_2 *— две задачи локального поиска. PLS-сведение задачи* Π_1 *к задаче* Π_2 *состоит в построении таких двух полиномиально вычислимых функций* h *и* g*, что:*

1) по произвольному входу x *задачи* Π_1 *функция* h *вычисляет некоторый вход* $h(x)$ *задачи* Π_2*;*

2) по произвольному решению s *для входа* $h(x)$ *функция* g *находит некоторое решение* $g(s, x)$ *для входа* x*;*

3) для всех $x \in \Pi_1$*, если* s *— локальный оптимум для входа* $h(x) \in \Pi_2$*, то* $g(s, x)$ *— локальный оптимум для входа* x*.*

Если такие функции h и g удается построить, то говорят, что задача Π_1 PLS-сводится к задаче Π_2. Понятие PLS-сводимости обладает свойством транзитивности: если Π_1 PLS-сводится к Π_2, а Π_2 PLS-сводится

к Π_3, то Π_1 PLS-сводится к Π_3. Если Π_3 полиномиально разрешима, то и Π_1 полиномиально разрешима. Далее будем говорить о сводимости, опуская обозначение PLS.

Определение 5 *Задачу* П *из класса PLS называют PLS-полной, если любая задача из класса PLS может быть сведена к ней.*

Полные задачи являются наиболее трудными в данном классе. Если хотя бы одна из них может быть решена за полиномиальное время, то и остальные задачи могут быть решены за полиномиальное время.

Первая PLS-полная задача *Circuit* была описана в [154]. Эта задача локального поиска формулируется для схемы x с m входами $(z_1, ..., z_m)$ и n выходами $(y_1, ..., y_n)$. Схема состоит из полиномиального числа элементов AND, OR, NOT. Множество ее допустимых решений $Sol(x)$ состоит из всех булевых векторов длины m. Целевая функция $F(z)$ определяется следующим образом:

$$F(z) = \sum_{j=1}^{n} 2^{j-1} y_j,$$

где y_j является j-м выходом схемы x. Окрестность $Flip(z)$ решения z состоит из всех булевых векторов длины m, имеющих расстояние Хэмминга от z равное 1. Максимизационная и минимизационная версии этой задачи являются PLS-полными [212]. PLS-полнота установлена для следующих задач локального поиска.

- Задача о максимальном разрезе с окрестностью $Flip$ ($Max-Cut$; $Flip$) [195]. Задан взвешенный неориентированный граф $G = (V, E)$, $w_e \geq 0$, $e \in E$. Допустимым решением является любое разбиение множества вершин на две части. Целевая функция — суммарный вес ребер, соединяющих две части разбиения. Окрестность $Flip$ состоит из решений, которые могут быть получены из данного переносом вершины из одной части разбиения в другую.

- Задача о выполнимости на максимум с окрестностью $Flip$ в двух постановках ($Max-2Sat$; $Flip$), ($Pos\,NAE\,Max-3Sat$; $Flip$) [165, 195].
  1. Задана булева функция в конъюнктивной нормальной форме. Каждая конъюнкция имеет неотрицательный вес и содержит не более двух переменных или их отрицаний. Требуется найти значения булевых переменных, при которых суммарный вес выполненных

конъюнкций был бы максимальным.
2. Заданы наборы, состоящие не более чем из трех булевых переменных без отрицаний. Набор считается выполненным, если не все значения его переменных совпадают. Каждому набору приписан неотрицательный вес. Требуется найти значения булевых переменных, при которых суммарный вес выполненных наборов был бы максимальным.

- Задача о разбиении множества вершин графа на две равные части с окрестностями *Swap*, Кернигана-Лина (KL) и их модификациями (*Graph Partitioning*; *Swap*, KL, FM, FM_1) [154, 195]. Задан взвешенный неориентированный граф $G = (V, E)$, $w_e \geq 0$, $e \in E$, с четным числом вершин $|V| = 2n$. Требуется найти разбиение множества V на два равномощных подмножества $V = V_1 \cup V_2, |V_1| = |V_2|$, для которого суммарный вес ребер, соединяющих вершины из V_1 с вершинами из V_2, был бы минимальным. Допустимым решением является любое разбиение V на два равномощных подмножества. Окрестность *Swap* состоит из решений, получаемых из заданного заменой одной вершины из V_1 на вершину из V_2. Окрестность KL строится с помощью следующей итерационной процедуры. На каждом шаге просматриваются все пары элементов $i_1 \in V_1, i_2 \in V_2$, каждый из которых еще не использовался на предыдущих итерациях, и выбирается наилучшая пара, то есть пара, которая максимально уменьшает целевую функцию, а если таких пар нет, то минимальным образом увеличивает ее. Если имеется несколько таких пар, то выбирается любая из них. Процедура завершается через n шагов и дает n соседних решений. При этом V_1 переходит в V_2, а V_2 переходит в V_1. Окрестность FM строится аналогично, но каждая итерация состоит из двух этапов [120]. На первом этапе просматриваются элементы из V_1, каждый из которых ранее не использовался, и выбирается наилучший. Он переводится в множество V_2. На втором этапе среди элементов множества V_2, каждый из которых ранее не использовался, выбирается наилучший и переводится в V_1. Наилучшим элементом по-прежнему считается элемент, перевод которого в другую долю максимально уменьшает целевую функцию, либо, если таких нет, минимальным образом увеличивает ее. Если имеется несколько наилучших элементов, то выбирается любой из них. Окрестность FM_1 содержит только одно соседнее решение, которое

получается на первой итерации указанной процедуры. Эта окрестность была введена в [195] и обозначалась *FM-Swap*.

- Задача о p-медиане с окрестностями $Swap, KL, FM, FM_1$ $(p-Median; FM_1, Swap, KL, FM)$ [40]. Задана $m \times n$ матрица $c_{ij} \geq 0$ и число p (медиана), $p < m$. Требуется найти подмножество строк S, $|S| = p$, для которого значение целевой функции $F(S) = \sum_{j=1}^{m} \min_{i \in S} c_{ij}$ было бы минимальным. Окрестности $Swap, KL, FM$ и FM_1 определяются по аналогии с предыдущей задачей.

- Простейшая задача размещения с окрестностью $Flip$ $(UFLP; Flip)$ [40]. Эта задача тесно связана с задачей о p-медиане. Задана $m \times n$ матрица $c_{ij} \geq 0$ и вектор $f_i \geq 0, i = 1, \ldots, m$. Требуется найти непустое подмножество строк S, для которого значение целевой функции $F(S) = \sum_{i \in S} f_i + \sum_{j=1}^{m} \min_{i \in S} c_{ij}$ было бы минимальным.

- Задача коммивояжера с окрестностями $k - opt, LK, LK'$ $(TSP; k - opt, LK, LK')$ [165, 180].

- Устойчивые конфигурации в нейросетях (модель Хорфилда) с окрестностью $Flip$ $(Stable\ Configuration; Flip)$ [195]. Задан полный взвешенный неориентированный граф $G = (V, E)$ с произвольными весами на ребрах $w_e, e \in E$ и вершинах $t_i, i \in V$. Конфигурация состоит в назначении каждой вершине i статуса $z_i \in \{-1, 1\}$. Вершину i называют устойчивой, если $z_i = 1$ и $\sum_{j \in V} w_{ij} z_i z_j + t_i \geq 0$ или $z_i = -1$ и $\sum_{j \in V} w_{ij} z_i z_j + t_i \leq 0$. Конфигурацию называют устойчивой, если все вершины являются устойчивыми. Задача состоит в нахождении устойчивой конфигурации. Введем целевую функцию $F(z) = \sum_{i \in V} t_i z_i + \sum_{(ij) \in E} w_{ij} z_i z_j$. Можно показать, что для неустойчивой вершины изменение ее статуса приводит к росту целевой функции. Фактически, устойчивые конфигурации соответствуют локальным максимумам для окрестности $Flip$.

На рис. 16 даны последовательности PLS-сведений для указанных PLS-полных задач. Приведенный список не является исчерпывающим.

Другие PLS-полные задачи, например, в области стратегических игр, можно найти в [114].

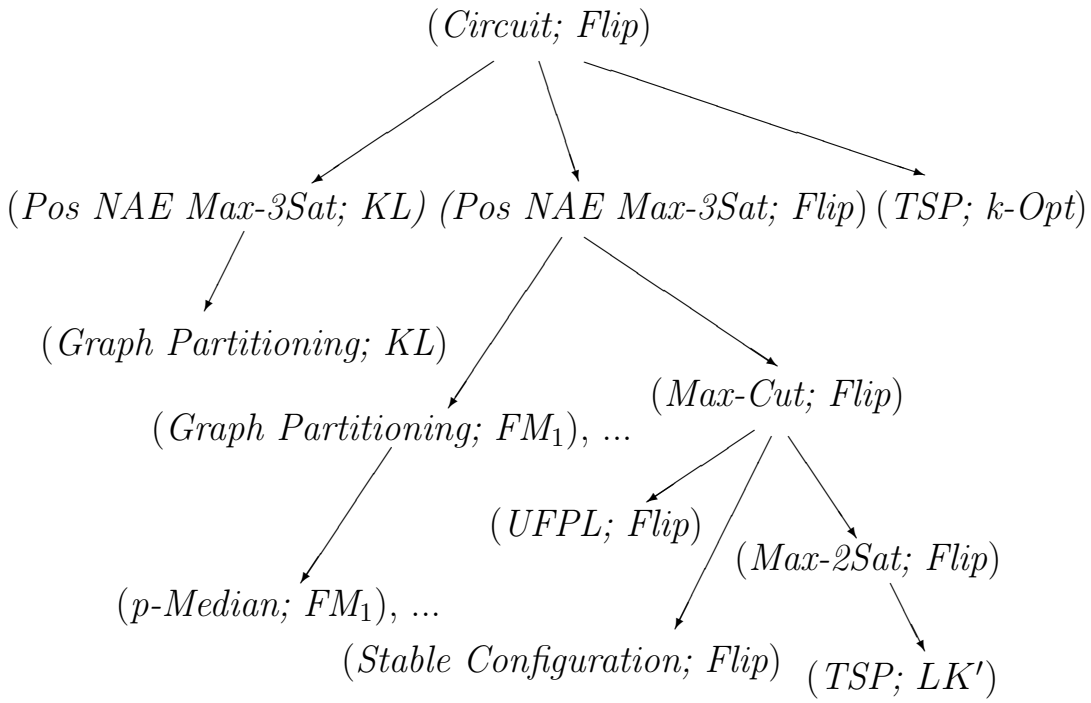


Рис. 16. *PLS-полные задачи*

По аналогии с классом P полиномиально разрешимых задач распознавания введен класс P_{PLS} полиномиально разрешимых задач локального поиска.

Определение 6 *Задача* $\Pi = (OP, N)$ *из класса PLS принадлежит классу* P_{PLS}*, если существует полиномиальный алгоритм, который по любому входу* x *задачи* Π *находит* N*-оптимальное решение.*

Как уже отмечалось выше, этот класс не является пустым. В частности, ему принадлежит задача линейного программирования с соответствующей окрестностью. Хотя симплекс-метод не является полиномиальным алгоритмом, существование метода эллипсоидов, который позволяет за полиномиальное время находить точное решение, влечет принадлежность задачи к классу P_{PLS}.

Современная теория сложности в значительной степени базируется на классах задач распознавания P и NP [25, 80]. С каждой оптимизационной задачей можно естественным образом связать соответствующую задачу распознавания, которая сводится к исходной задаче: для данного входа

найти такое решение, что значение целевой функции не превосходит заданной величины. Алгоритм решения оптимизационной задачи можно использовать для решения данной задачи распознавания. Поэтому классов P и NP обычно достаточно для изучения сложности оптимизационных задач, когда нас интересует нахождение оптимальных решений. Для задач локального поиска неизвестно таких естественных сводимостей, связывающих их с задачами распознавания. Рассмотрим, например, следующую задачу распознавания: существует ли локальный оптимум, на котором значение целевой функции не превосходит заданной величины? Неизвестно, как использовать алгоритм решения задачи локального поиска для того, чтобы найти ответ на эту задачу распознавания. По сути рассматриваемая задача распознавания представляется более сложной, чем задача нахождения какого-нибудь локального оптимума. Если эта или подобные задачи распознавания являются NP-полными, то это не означает, что задача локального поиска является трудноразрешимой.

В [154] введены классы P_S и NP_S — формальные аналоги классов задач поиска, решаемых за полиномиальное время на детерминированных и недетерминированных машинах Тьюринга соответственно. Напомним, что класс NP_S состоит из бинарных отношений $R \subseteq \{0,1\}^* \times \{0,1\}^*$, которые

— полиномиально ограничены, т. е. если $(x, y) \in R$, то длина слова y полиномиально ограничена длиной слова x;

— полиномиально распознаваемы, т. е. существует полиномиальный алгоритм, который для любой пары (x, y) проверяет, принадлежит ли она отношению R или нет. Задача поиска, связанная с отношением R, заключается в том, чтобы для любого x найти такой y, что $(x, y) \in R$. Эта задача содержится в классе P_S, если существует полиномиальный алгоритм, который для любого x либо находит y, такой что $(x, y) \in R$, либо сообщает об отсутствии такого y.

Между классом PLS и классами P_S и NP_S установлена следующая цепочка вложений

$$P_S \subseteq PLS \subseteq NP_S.$$

Первое включение означает, что любая полиномиально разрешимая задача поиска представима в виде задачи из класса PLS. Второе включение говорит о том, что любая задача из класса PLS представима в виде задачи поиска, решаемой за полиномиальное время на недетерминированной машине Тьюринга. Известно [212], что данное включение является строгим, если NP≠co-NP.

Покажем, что класс P_{PLS} эквивалентен классу P_S. Пусть задача локального поиска П принадлежит классу P_{PLS}, а входы и решения задачи П являются двоичными словами. Тогда бинарное отношение

$$R_{\Pi} = \{(x, y) \mid y - \text{ локальный оптимум для входа } x \text{ задачи } \Pi\}$$

полиномиально ограничено, так как в классе PLS содержатся только такие задачи, для которых длина записи любого допустимого решения ограничена полиномом от длины записи исходных данных. Полиномиальная распознаваемость этого отношения следует из полиномиальности алгоритма C в определении класса PLS. Принадлежность отношения R_{Π} классу P_S следует из существования полиномиального алгоритма для задачи П. Таким образом, $\mathrm{P}_{PLS} \subseteq \mathrm{P}_S$.

Покажем обратное включение. Рассмотрим произвольное отношение R из класса P_S. Соответствующую задачу поиска представим в виде задачи из класса P_{PLS}. Рассмотрим произвольное двоичное слово x. По определению за полиномиальное время можно узнать, существует ли такое y, что $(x, y) \in R$. Если нет, то будем считать, что x не является входом задачи локального поиска. Если да, то положим:

множество допустимых решений $Sol(x) = \{y\}$,

целевая функция $F(y, x) = 0$,

окрестность $N(y, x) = \{y\}$.

Такая задача локального поиска удовлетворяет определению 3, т.е. $\mathrm{P}_S \subseteq \mathrm{P}_{PLS}$.

Завершая обсуждение классов PLS, P_{PLS}, P_S и NP_S заметим, что в настоящее время нет прямых или косвенных аргументов в пользу совпадения классов PLS и P_{PLS}. Таким образом, наряду с центральной проблемой теории сложности $\mathrm{P}=^{?}\mathrm{NP}$ существует аналогичная проблема $\mathrm{P}_{PLS} =^{?} \mathrm{PLS}$ о локальных оптимумах. Если $\mathrm{P}_{PLS} \neq \mathrm{PLS}$, то $\mathrm{P} \neq \mathrm{NP}$. Может быть, доказать первое неравенство легче, чем второе. В любом случае, вопрос о совпадении классов P_{PLS} и PLS является интересным и актуальным.

4.3 PLS-полные задачи размещения

В данном разделе рассматриваются простейшая задача размещения, задача о p-медиане и их обобщения. Будет показана PLS-полнота поиска локальных минимумов для этих задач при различных полиномиально проверяемых окрестностях.

Пусть $S \subseteq I$, $|S| = p$. Рассмотрим следующие окрестности.

1. Окрестность $Flip(S)$ образуют все подмножества $S' \subseteq I$, которые получаются из S удалением одного элемента из S или добавлением одного элемента из $I \setminus S$. Окрестность содержит ровно $|I|$ элементов.

2. Окрестность $Swap(S)$ образуют все подмножества $S' \subset I$ мощности p, которые получаются из S заменой одного элемента из S на элемент из дополнения $\overline{S} = I \setminus S$. Окрестность содержит $p(|I| - p)$ элементов.

3. Окрестность $LK(S)$ (Lin-Kernighan) [156, 157]. Она строится с помощью следующей итерационной процедуры. На каждом шаге просматриваются все пары элементов $i_1 \in S, i_2 \in \overline{S}$, каждый из которых ранее не использовался в перестановках, и выбирается наилучшая пара, т.е. пара, которая максимально уменьшает целевую функцию, либо, если таких нет, минимальным образом увеличивает её. Если таких пар несколько, то выбирается любая из них. Процедура останавливается через k шагов, $1 \leq k \leq \min\{p, |I| - p\}$, и дает k соседних решений для S.

4. Окрестность $LK_1(S)$ состоит из одного подмножества, которое получается на первом шаге предыдущей процедуры. По построению эта окрестность является частью окрестности $Swap$.

5. Окрестность $FM(S)$ (Fiduccia-Mattheyses) [120]. Она строится аналогично окрестности $LK(S)$. Каждый шаг этой процедуры состоит из двух этапов. На первом этапе просматриваются элементы из S, каждый из которых ранее не использовался, и выбирается наилучший. Он переводится в дополнение множества S. На втором этапе среди элементов множества $\overline{S}$, каждый из которых ранее не использовался, выбирается наилучший и переводится в S. Наилучшим элементом по-прежнему считается элемент, перевод которого в другое множество максимально уменьшает целевую функцию, либо, если таких нет, минимальным образом увеличивает ее. Если имеется несколько наилучших элементов, то выбирается любой из них. Окрестность содержит k допустимых решений, $1 \leq k \leq \min\{p, |I| - p\}$.

6. Окрестность $FM_1(S)$ состоит из одного подмножества, которое получается на первом шаге предшествующей процедуры. Данная окрестность была введена для задачи о разбиении графа [195] и обозначалась через $FM\text{-}Swap$.

7. Окрестность $k\text{-}Swap(S)$ [210] состоит из подмножеств, которые получаются из S следующим образом. Выбирается не более k пар (i_1, i_2), $i_1 \in S$, $i_2 \in \overline{S}$, не содержащих общих элементов, и для каждой пары производится замена элемента i_1 на элемент i_2. Окрестность имеет

$O(p^k(|I|-p)^k)$ элементов и является полиномиальной при любом фиксированном k.

Легко проверить, что задача о p-медиане и простейшая задача размещения с соответствующими окрестностями принадлежат классу PLS. Ниже будут показаны связи между этими задачами локального поиска, но предварительно установим отношения предшествования между введёнными окрестностями.

Лемма 4 *Справедливы следующие соотношения между окрестностями:*

$$FM_1 \preceq Swap \preceq LK_1 \preceq LK,$$
$$LK_1 \preceq Swap \preceq k\text{-}Swap,$$
$$FM_1 \preceq FM.$$

Доказательство легко следует из определения окрестностей. Таким образом, окрестность FM_1 является самой слабой, окрестности $Swap$ и LK_1 эквивалентны, а окрестности LK, FM, $k-Swap$ несравнимы. Заметим, что окрестность $Flip$ не сравнима ни с одной из приведенных выше. Часто она используется либо независимо от данных окрестностей, либо в сочетании с некоторыми из них, например, $Flip \cup Swap$ [132].

Лемма 5 *Пусть задачи* $\Pi_1 = (OP, N_1)$, $\Pi_2 = (OP, N_2)$ *принадлежат классу PLS и* $N_1 \preceq N_2$. *Тогда* Π_1 *сводится к* Π_2.

Доказательство. Так как из N_2-оптимальности решения следует его N_1-оптимальность, то в качестве функций h и g из определения PLS–сводимости возьмём тождественные функции. Их полиномиальная вычислимость очевидна. □

Следствие 3 *Задача* $(p\text{-}median, Swap)$ *эквивалентна задаче* $(p\text{-}median, LK_1)$ *и сводится к задачам* $(p\text{-}median, LK)$ *и* $(p\text{-}median, k\text{-}Swap)$. *Задача* $(p\text{-}median, FM_1)$ *сводится к задачам* $(p\text{-}median, FM)$, $(p\text{-}median, Swap)$, $(p\text{-}median, LK_1)$, $(p\text{-}median, LK)$, $(p\text{-}median, k\text{-}Swap)$.

Теорема 19 *Задача поиска локального минимума для простейшей задачи размещения с окрестностью Flip является PLS-полной.*

Доказательство. Рассмотрим задачу о максимальном разрезе *Max-Cut*. Задан неориентированный граф $G = (V, E)$ с неотрицательными весами ребер $w_e \geq 0, e \in E$. Допустимым решением задачи является любое

разбиение вершин на два подмножества V_1 и V_2, т.е. разрез. В качестве целевой функции используется вес разреза

$$W(V_1, V_2) = \sum(w_e \mid e = (i_1, i_2) \in E, i_1 \in V_1, i_2 \in V_2).$$

Окрестность *Flip* состоит из всех разрезов, получающихся из заданного переносом одной вершины в другое подмножество. Задача состоит в поиске разреза, являющегося локальным максимумом для окрестности *Flip*.

Покажем PLS-сведение данной задачи локального поиска к задаче $(UFLP, Flip)$. Для этого следует предъявить функции h, g из определения PLS сводимости с требуемыми свойствами. Обозначим через $E(i)$ множество ребер в G, инцидентных вершине $i \in V$.

Положим $I = V$, $J = E$,

$$f_i = \sum_{e \in E(i)} w_e, \ i \in I,$$

$$c_{ie} = \begin{cases} 0, & \text{если } i = i_1 \text{ или } i = i_2 \text{ при } e = (i_1, i_2), \\ 2w_e & \text{в противном случае,} \end{cases} \quad i \in I, e \in J.$$

Для любого решения $S \subseteq I$ определим разрез по следующему правилу: $V_1 = S$, $V_2 = V \setminus V_1$. Проверим равенство

$$\sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} c_{ij} + W(V_1, V_2) = 2 \sum_{e \in E} w_e,$$

которое гарантирует требуемые свойства предлагаемого PLS сведения. Рассмотрим три возможных случая для ребра $e = (i_1, i_2) \in E$.

1) $i_1, i_2 \in V_1$. Вес w_e входит в качестве слагаемого в величины f_{i_1}, f_{i_2} и не включается в величину разреза $W(V_1, V_2)$. Кроме того, $\min_{i \in S} c_{ie} = 0$. Следовательно, величина w_e дважды представлена в обеих частях равенства.

2) $i_1, i_2 \in V_2$. Величины f_{i_1}, f_{i_2} не включаются в первую сумму левой части равенства. Более того, величина разреза $W(V_1, V_2)$ не содержит слагаемого w_e. Однако $\min_{i \in S} c_{ie} = 2w_e$. Следовательно, величина w_e снова дважды представлена в обеих частях равенства.

3) $i_1 \in V_1, i_2 \in V_2$. Вес w_e входит в качестве слагаемого в величину f_{i_1} и включается в разрез $W(V_1, V_2)$. Кроме того, $\min_{i \in S} c_{ie} = 0$. Величина w_e опять дважды представлена в обеих частях равенства. □

Следствие 4 *Задача поиска локальноro минимума относительно окрестности $Flip$ является PLS–полной для любого обобщения простейшей задачи размещения. Утверждение остается верным и для любой другой полиномиально проверяемой окрестности, включающей в себя окрестность $Flip$.*

Доказательство. Если задача является обобщением простейшей задачи размещения, значит, легко построить функцию h, которая будет указывать, каким именно частным случаем является простейшая задача размещения для исходной задачи. Аналогично строится и функция g. Для завершения доказательства остается только заметить, что получив локальный минимум по более широкой окрестности, автоматически получаем локальный минимум и для окрестности $Flip$. □

При формулировке и доказательстве следующих результатов используется задача о разбиении множества вершин графа на два подмножества равной мощности (*Max-Graph Partitioning* или кратко GP). Приведём её постановку. Задан граф $G = (V, E)$ с чётным числом вершин $|V| = 2n$ и весами рёбер $w_e \geq 0, e \in E$. Допустимым решением является любое разбиение множества V на два подмножества V_1, V_2 равной мощности. Требуется найти допустимое решение с максимальным весом разреза $W(V_1, V_2)$, т. е. с максимальной суммой весов рёбер, у которых один конец содержится в множестве V_1, а другой — в множестве V_2.

В дальнейшем рассматривается задача нахождения локального оптимума для задачи GP с окрестностью FM_1, которая определяется так же, как для задачи о p-медиане. Плотная PLS-полнота задачи (GP, FM_1) доказана в [195, 212].

Для доказательства полноты задачи о p-медиане с любой из введённых выше окрестностей достаточно этот факт установить для задачи с окрестностью FM_1. В следующей теореме фактически устанавливается более сильный результат, который впоследствии позволит оценить снизу число шагов алгоритма улучшения для ряда задач локального поиска.

Теорема 20 *Задача (GP, FM_1) сводится к задаче $(p\text{-}median, FM_1)$.*

Доказательство. По исходным данным задачи GP построим исходные данные задачи о p-медиане так, чтобы любому допустимому разрезу (V_1, V_2) соответствовал набор из p предприятий и сумма значений целевых функций на этих решениях была постоянной.

Обозначим через E_i множество рёбер в графе G, инцидентных вершине $i \in V$, и пусть $W_i = \sum\limits_{e \in E_i} w_e,\ W = \sum\limits_{e \in E} w_e$.

Положим $I = \{1, \ldots, |V|\},\ J = \{1, \ldots, |E| + |V|\},\ p = n = |V|/2$. Каждому $j = 1, \ldots, |E|$ поставим в соответствие ребро $e \in E$ и положим

$$c_{ij} = \begin{cases} 0, & \text{если } \ e = (i_1, i_2), (i = i_1) \vee (i = i_2), \\ 2w_e & \text{в противном случае,} \end{cases}$$

а для каждого $j = |E| + 1, \ldots, |E| + |V|$ положим

$$c_{ij} = \begin{cases} 0, & \text{если } \ i = j - |E|, \\ W - W_i & \text{в противном случае.} \end{cases}$$

Разрезу (V_1, V_2) поставим в соответствие решение $S = V_1$ и проверим, что

$$\sum_{j \in J} \min_{i \in S} c_{ij} + W(V_1, V_2) = nW.$$

Заметим, что

$$\sum_{j=1}^{|E|} \min_{i \in S} c_{ij} = 2 \sum (w_e \mid e = (i_1, i_2), i_1, i_2 \notin S)$$

и

$$\sum_{j=1+|E|}^{|J|} \min_{i \in S} c_{ij} = \sum_{i \notin S} (W - W_i) = nW - \sum_{i \notin S} W_i.$$

Так как величина $\sum\limits_{i \notin S} W_i$ содержит вес каждого ребра из разреза (V_1, V_2) ровно один раз, а вес ребра $e = (i_1, i_2), i_1, i_2 \notin V_1$ включается в нее дважды, то

$$\sum_{i \notin S} W_i = W(V_1, V_2) + \sum_{j=1}^{|E|} \min_{i \in S} c_{ij}.$$

Отсюда следует утверждение теоремы. □

Следствие 5 *Для полноты задачи* $(p\text{-}median, N)$ *из класса PLS достаточно, чтобы окрестность* N *была не слабее окрестности* FM_1.

Покажем, как следствие 5 может быть использовано для анализа сложности задач локального поиска. Из леммы 4 и следствия 5 имеем, что задача о p-медиане с окрестностями $Swap$, LK, LK_1, FM, $k-Swap$ является полной. Рассмотрим одну из древесных окрестностей, которая может быть определена следующим образом. Она включает в себя окрестность $Swap$, и к каждому элементу этой окрестности добавляется цепочка соседних решений по правилам окрестности LK (рис. 17). Обозначим новую окрестность через $TreeLK$.

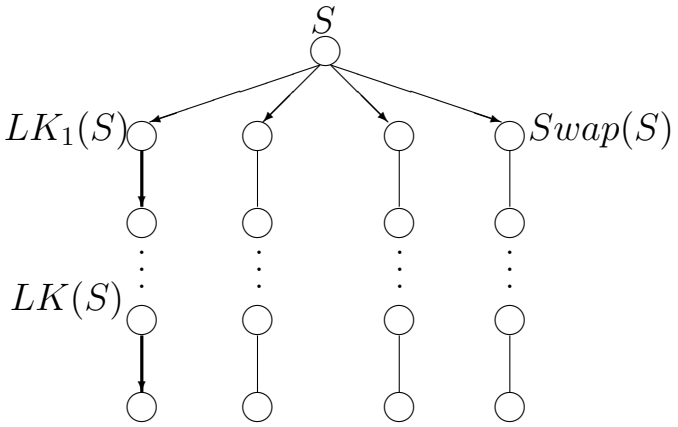


Рис 17. *Окрестность* $TreeLK(S)$

Она является полиномиально проверяемой, и $TreeLK$-оптимальное решение является $Swap$-оптимальным. Отсюда следует PLS полнота задачи $(p-median, TreeLK)$. Другие примеры древесных окрестностей можно найти в [168].

Следствие 6 *Если* $P_{PLS} \neq PLS$ *и задача о p-медиане с окрестностью* N *полиномиально разрешима, то* $FM_1 \not\preceq N$.

Доказательство. Если $FM_1 \preceq N$, то задача $(p-median, N)$ будет полной и, следовательно, $P_{PLS} = \text{PLS}$. Таким образом, если N — полиномиально проверяемая окрестность, то для полиномиальной разрешимости задачи $(p-median, N)$ необходимо, чтобы эта окрестность была либо слабее окрестности FM_1, либо эти окрестности были несравнимы. Первое означает, что множество FM_1-оптимальных решений должно быть собственным подмножеством множества N-оптимальных решений. Второе означает, что из FM_1-оптимальности решения не должна следовать N-оптимальность и наоборот. □

Аналогичные утверждения можно сформулировать для других известных оптимизационных задач: задачи коммивояжера, задачи о максимальном разрезе, о разбиении графа и другие [212].

4.4 Временная сложность локального спуска

Принадлежность задачи локального поиска к классу PLS гарантирует возможность применения стандартного алгоритма локального спуска, каждый шаг которого является полиномиальным. Ниже будет показано, что для многих известных комбинаторных задач с естественно заданными окрестностями алгоритму локального спуска требуется в худшем случае экспоненциальное число шагов для достижения локального минимума.

Определение 7 *Графом переходов* $TG_{\Pi}(x)$ *для входа* x *задачи* Π *называется ориентированный граф с множеством вершин* $Sol(x)$ *и множеством дуг вида* (s, s'), *где* $s' \in N(s, x)$ *и* $F(s', x) < F(s, x)$. *Высота вершины* s *есть длина кратчайшего пути в графе* $TG_{\Pi}(x)$ *из вершины* s *в сток (т.е. в локальный минимум). Высота графа* $TG_{\Pi}(x)$ *определяется как максимальная высота его вершин.*

Из определения следует, что граф переходов является ациклическим и может иметь экспоненциальное число вершин относительно длины входа. Если s не является локальным минимумом, то переходу от s к s' в графе $TG_{\Pi}(x)$ соответствует ребро (s, s'). Применяя алгоритм локального спуска к разным начальным решениям, будем получать различные пути, ведущие из начальных вершин в стоки. Высота вершины по сути есть оценка снизу на число шагов для достижения локального минимума. Эта оценка не зависит от применяемого правила замещения. Таким образом, алгоритму требуется экспоненциальное число шагов при любом правиле замещения тогда и только тогда, когда найдутся исходные данные рассматриваемой задачи, для которых высота графа переходов является экспоненциальной функцией от длины записи исходных данных.

Определение 8 *Пусть задача* Π_1 *сводится к задаче* Π_2 *и* h, g *— соответствующие функции. Будем говорить, что эта сводимость является плотной, если для любого входа* x *задачи* Π_1 *можно указать*

подмножество R допустимых решений входа $y = h(x)$ задачи Π_2 со следующими свойствами.

1. R содержит все локальные минимумы для входа y.

2. Существует полиномиальный алгоритм, который для каждого решения p входа x позволяет построить решение $q \in R$ для входа y такое, что $g(q, x) = p$.

3. Пусть граф переходов $TG_{\Pi_2}(y)$ содержит ориентированный путь из вершины $q \in R$ в вершину $q' \in R$ такой, что нет промежуточных вершин из R, и пусть $p = g(q, x)$, $p' = g(q', x)$ — соответствующие решения для входа x. Тогда $p = p'$ или граф переходов $TG_{\Pi_1}(x)$ содержит дугу из вершины p в вершину p'.

Рассмотрим произвольный путь T в графе $TG_{\Pi_2}(y)$ с начальной и конечной вершинами из множества R. Возьмем образы вершин этого пути в графе $TG_{\Pi_1}(x)$ относительно отображения g. Условие 3 гарантирует, что участку пути T между любыми двумя вершинами q, q' из множества R (не содержащему промежуточных вершин из этого же множества) соответствует либо вершина, являющаяся образом q и q', либо ребро, соединяющее образы этих вершин. Следовательно, пути T можно сопоставить путь $T(g)$ в графе $TG_{\Pi_1}(x)$, вершины которого являются образами вершин из T. При этом длина $T(g)$ не превосходит длины исходного пути. Другими словами, отображение g является сжимающим относительно множества R, так как оно переводит любой путь из графа $TG_{\Pi_2}(y)$, начальная и конечная вершины которого лежат в R, в путь не большей длины в графе $TG_{\Pi_1}(x)$.

С содержательной точки зрения, введение множества R в данном определении дает возможность переводить произвольный путь в графе $TG_{\Pi_2}(y)$, содержащийся в R и ведущий в сток, в путь не большей длины в графе $TG_{\Pi_1}(x)$, также ведущий в некоторый сток.

Пусть p – произвольная вершина графа $TG_{\Pi_1}(x)$. Возьмем ее прообраз $q = g^{-1}(p)$ из множества R. Рассмотрим один из путей минимальной длины, выходящий из вершины q и ведущий в некоторый сток графа $TG_{\Pi_2}(y)$. В графе $TG_{\Pi_1}(x)$ ему соответствует путь не большей длины, ведущий из вершины p в один из стоков этого графа. Таким образом, высота вершины q не меньше высоты вершины p в графе $TG_{\Pi_1}(x)$. То есть высота графа $TG_{\Pi_2}(y)$ не меньше высоты графа $TG_{\Pi_1}(x)$. Следовательно, плотная сводимость сохраняет нижние оценки на число шагов алгоритма локального спуска.

Определение 9 *Задача* П *из класса PLS называется плотно полной, если все задачи из этого класса плотно сводятся к ней.*

Все PLS-полные задачи, изображенные на рис. 16, являются плотно PLS-полными [212]. В частности, плотно PLS полными являются и рассмотренные в предыдущем разделе дискретные задачи размещения.

Теорема 21 *Задача локального поиска $(UFLP, Flip)$ является плотно PLS-полной.*

Доказательство. Вернемся к доказательству теоремы о PLS-полноте задачи ($UFLP$, $Flip$) и убедимся, что предъявленное сведение является плотным. Так как задача *(Max-Cut, Flip)* является плотно PLS-полной задачей, то отсюда и будет следовать утверждение теоремы.

Возьмем в качестве множества R все множество допустимых решений простейшей задачи размещения. Заметим, что графы переходов при выбранных функциях h и g имеют одинаковое число вершин. Так как сумма целевых функций задач $UFLP$ и *Max-Cut* является константой, то на любых соседних решениях увеличение одной целевой функции влечет уменьшение другой. Вспоминая, что одна задача является задачей на максимум, а другая — на минимум, приходим к выводу, что графы переходов этих задач совпадают. Следовательно, их высоты тоже совпадают, и построенное сведение является плотным. □

Сформулируем достаточное условие плотной сводимости двух задач из класса PLS.

Лемма 6 *Пусть задачи* $\Pi_1 = (OP, N_1)$, $\Pi_2 = (OP, N_2)$ *принадлежат классу PLS и* $N_1 \preceq N_2$. *Если для каждого входа x оптимизационной задачи OP граф $TG_{\Pi_2}(x)$ является подграфом графа $TG_{\Pi_1}(x)$, то задача* Π_1 *плотно сводится к задаче* Π_2.

Доказательство. По лемме 5 задача Π_1 сводится к задаче Π_2. Покажем, что это сведéние является плотным. Возьмём в качестве функций h и g тождественные функции. В качестве R возьмем множество всех допустимых решений. Условия 1 и 2 определения 8 проверяются тривиально. По условию граф $TG_{\Pi_2}(x)$ является подграфом графа $TG_{\Pi_1}(x)$. Поэтому любой путь в графе $TG_{\Pi_2}(x)$ является путём в графе $TG_{\Pi_1}(x)$, а каждый сток (локальный минимум) в графе $TG_{\Pi_2}(x)$ является стоком в графе $TG_{\Pi_1}(x)$. Следовательно, выполняется условие 3 определения

плотной сводимости. □

Следствие 7 *Задача* $(p\text{-}median, Swap)$ *плотно сводится к задаче* $(p\text{-}median, LK_1)$.

Так как граф переходов в задаче с окрестностью LK_1 является подграфом графа переходов в задаче с окрестностью $Swap$, то утверждение следует из леммы 6. □

Теорема 22 *Задача локального поиска* $(p-median, FM_1)$ *является плотно PLS-полной.*

Доказательство. Возьмем в качестве множества R все множество допустимых решений задачи о p-медиане. Повторяя рассуждения при доказательстве предыдущей теоремы, получаем совпадение графов переходов задач $(p - median, Swap)$ и (GP, FM_1), откуда и следует требуемое. □

Следующее утверждение показывает, что можно получить плотную PLS-полноту задачи о p-медиане с каждой из окрестностей $Swap, LK$, LK_1, FM.

Теорема 23 *Задача о* p*-медиане с каждой из окрестностей* $Swap, LK$, LK_1, FM, FM_1 *является плотно полной.*

Доказательство. Задача GP с каждой из указанных окрестностей, кроме окрестности LK_1, является плотно полной [195, 212]. Сводимость теоремы 20 такова, что существует взаимно однозначное соответствие между допустимыми разбиениями задачи GP и допустимыми решениями некоторого частного случая задачи о p-медиане. Значения целевых функций на соответствующих решениях в сумме дают постоянную величину. Таким образом, граф переходов задачи GP с любой из рассматриваемых окрестностей изоморфен графу переходов этого частного случая задачи о p-медиане с той же окрестностью. Отсюда следует утверждение теоремы.

Для окрестности LK_1 утверждение теоремы следует из доказанной уже плотной полноты задачи $(p - median, Swap)$ и следствия 5. □

Обратимся теперь к вопросу о временной сложности алгоритмов локального улучшения. Чтобы показать, что в худшем случае такому алгоритму потребуется экспоненциальное число шагов для плотно PLS-полных задач, достаточно найти в классе PLS хотя бы одну задачу, обладающую этим свойством.

Лемма 7 *[195] В классе PLS существует задача, для которой алгоритму локального спуска требуется экспоненциальное число шагов при любом правиле замещения.*

Доказательство этой леммы является конструктивным, и соответствующая задача локального поиска имеет следующую простую структуру. Для любого входа x длины n множество допустимых решений $Sol(x)$ состоит из всех n-мерных векторов, занумерованных от 0 до 2^n-1. Каждому решению i приписано значение целевой функции, равное её номеру i. Если $i > 0$, то у решения i есть только один сосед $i-1$. Поэтому в задаче имеется только один локальный (глобальный) минимум $i = 0$. Граф переходов $TG_{\Pi}(x)$ является путем длины $2^n - 1$. Алгоритм локального спуска, начиная с вершины $i = 2^n-1$, выполнит экспоненциальное число шагов. Таким образом, получаем следующее утверждение.

Теорема 24 *В худшем случае для задачи $UFLP$ с окрестностью $Flip$ и задачи $p-median$ с окрестностями FM_1, $Swap, LK$, LK_1, FM алгоритму локального спуска потребуется экспоненциальное число шагов для достижения локального минимума при любом правиле замещения.*

Для задачи о раскраске вершин графа в два цвета в [210] приводится семейство графов, для которого алгоритм наискорейшего спуска с окрестностью $Flip$ требует экспоненциального числа итераций. В [40] аналогичное семейство приводится для задачи о p–медиане с окрестностью LK_1.

Рассмотрим теперь обобщение простейшей задачи размещения, например, задачу выбора состава системы технических средств при многоэтапном процессе выполнения работ, динамическую задачу выбора оптимального состава системы и др. Так как задача $UFLP$ является их частным случаем, то аналог предыдущей теоремы имеет место и для этих задач. Указанное замечание относится и к обобщениям задачи $p-median$. К сожалению, полученные нижние оценки на число шагов алгоритмов локального улучшения нельзя столь же просто перенести на случай более

мощных окрестностей, например, $k - Swap$ или $k - Flip$ для заданной константы k. Расширение окрестности влечет появление новых дуг в графе переходов. Это может сократить путь из вершины в сток и привести к уменьшению высоты вершины. С другой стороны, расширение окрестности может повлечь за собой сокращение числа стоков (локальных оптимумов) и, следовательно, увеличить кратчайший путь до ближайшего стока и высоту вершины. Таким образом, расширение окрестности имеет противоречивые последствия. Вопрос о нижних оценках в этом случае требует отдельного рассмотрения. В следующей главе будут исследоваться игровые модели размещения, в которых несколько лиц независимо друг от друга принимают решения об открытии предприятий. Для такой игры можно построить соответствующую оптимизационную задачу, где каждому локальному оптимуму будет соответствовать равновесие по Нэшу в исходной игре. В этом случае удается доказать экспоненциальные нижние оценки для числа шагов алгоритмов локального улучшения с более мощными окрестностями, чем $Flip$ или $Swap$, например, для $k - Swap$, $k - Flip$ и их объединением. Однако для исходных задач размещения $UFLP$, $p - median$ и их обобщений данный вопрос остается открытым.

Обратимся теперь к более сложной задаче локального поиска с фиксированной начальной точкой: для заданных входа, окрестности и начального решения найти локальный минимум, который может быть получен алгоритмом локального спуска из данного начального решения. Такую задачу называют стандартной задачей локального поиска. Раньше требовалось найти любой локальный минимум, теперь надо найти локальный минимум, достижимый из заданного начального решения. Стандартная задача локального поиска лежит в классе PSPACE. Это означает, что объем требуемой памяти для реализации алгоритма ограничен полиномом. Действительно, по предположению длина начального решения, как и всякого допустимого решения, ограничена полиномом от длины входа. Нахождение лучшего соседа, если таковой существует, осуществляется за полиномиальное время по определению задачи из класса PLS и, следовательно, требует полиномиально ограниченной памяти. Так как не нужно хранить все промежуточные решения, то требуемое пространство всегда полиномиально ограничено.

Известно [195], что если задача локального поиска Π_1 плотно сводится к задаче Π_2, то существует полиномиальное сведéние стандартной задачи локального поиска для Π_1 к стандартной задаче локального поиска для

Π_2. Кроме того, в классе PLS существует задача П, для которой соответствующая стандартная задача локального поиска является PSPACE-полной [195]. Из этих двух утверждений получаем следующее.

Теорема 25 *Для простейшей задачи размещения с окрестностью Flip и задачи о p-медиане с окрестностями Swap, LK, LK_1, FM, FM_1 соответствующие стандартные задачи локального поиска являются PSPACE-полными.*

Рассмотрим теперь ограниченную версию данной задачи. Для заданного начального решения требуется ответить на вопрос: *существует ли локальный минимум, достижимый из этого решения с помощью алгоритма локального спуска за заданное число итераций?* Заметим, что это задача лежит в классе NP, т.к. ответ "ДА", предъявляемый в виде ориентированного пути в графе переходов, может быть проверен за полиномиальное время. Установлено [124], что для ряда известных комбинаторных задач соответствующая задача поиска локального оптимума является NP–полной. Аналогичный вопрос для дискретных задач размещения остается пока открытым.

4.5 Локально седловые точки

Вероятностные методы локального поиска [98] показывают высокую эффективность при решении NP-трудных задач. Их гибкость в адаптации к сложным математическим моделям открывает широкую дорогу к практическому использованию. Несмотря на различие концепций, эти методы часто используют процедуры поиска локальных оптимумов. Концентрация внимания на локальных оптимумах близка в идейном смысле к классическому понятию вариации в теории экстремальных задачи. Тем не менее, необходимые условия Куна–Такера (КТ), подразумевающие взятие производных, в методах комбинаторной оптимизации не используются. Создается впечатление, что классические методы непрерывной оптимизации слишком далеки от комбинаторных. Однако в ряде случаев условие локальной оптимальности дискретного решения эквивалентно условиям КТ для непрерывной задачи, полученной релаксацией требования целочисленности переменных [75]. Для NP-трудных задач проблема часто состоит не в том, чтобы найти локальные оптимумы, а в том, что их оказывается слишком много. Найти среди них точку глобального оптимума представляется серьезной проблемой, что, по-видимому, и делает

исходную задачу NP-трудной. Понятие глобального оптимума не является конструктивным. Даже получив его, доказать оптимальность очень трудно. Аналогичные проблемы возникают и в непрерывной оптимизации. Там необходимые условия носят локальный характер. Они опираются на понятие вариации. В комбинаторной оптимизации реализуется та же идея, и понятие окрестности играет здесь центральную роль.

Выбор окрестности, как уже отмечалось выше, является важным при построении алгоритмов локального поиска. От него существенно зависит трудоемкость одного шага алгоритма, общее число шагов и, в конечном счете, качество получаемых локальных оптимумов. На сегодняшний день нет и, возможно, никогда не будет единого правила выбора окрестности. Для каждой задачи окрестность приходится определять заново, учитывая специфику данной задачи. Более того, по-видимому, для каждой задачи можно предложить несколько окрестностей разной мощности и, как следствие, получить разные множества локальных оптимумов. Ниже будет показано, что для задачи $p - median$ и ее обобщений, соответствующих задаче минимизации полиномов от булевых переменных, поиск локальных минимумов относительно окрестности $Swap$ равносилен выделению локальных седловых точек функции Лагранжа. Такие седловые точки удовлетворяют условиям Куна–Такера, что в свою очередь позволяет глубже понять связь методов локального поиска и мотивацию к их использованию при решении дискретных задач размещения. Расширение окрестности приводит к сокращению числа локальных оптимумов, но не нарушает указанного свойства, что дает возможность сократить число претендентов на глобальный оптимум и повысить эффективность численных методов.

Рассмотрим следующую задачу минимизации полинома от булевых переменных

$$P(z) = \sum_{i \in I} \sum_{j \in J} \nabla c_{ij} \prod_{k \in S_{ij}} z_k,$$

при ограничениях:

$$\sum_{i \in I} z_i = m - p,$$

$$z_i \in \{0, 1\}, i \in I.$$

Если $\nabla c_{ij} \geq 0$, то эта задача эквивалентна задаче о p–медиане: найти

$$\min \sum_{j \in J} \sum_{i \in I} c_{ij} x_{ij}$$

при ограничениях:

$$\sum_{i\in I} x_{ij} = 1, \qquad j \in J,$$
$$x_{ij} \le y_i, \qquad i \in I, j \in J,$$
$$\sum_{i\in I} y_i = p,$$
$$y_i, x_{ij} \in \{0,1\}, \quad i \in I, j \in J.$$

При произвольных ∇c_{ij} эта задача эквивалентна задаче о p– медиане с предпочтениями клиентов ($BPMP$): найти минимум функции

$$F(y) = \sum_{j\in J}\sum_{i\in I} c_{ij} x^*_{ij}(y_i)$$

при ограничениях:

$$\sum_{i\in I} y_i = p,$$
$$y_i \in \{0,1\}, \quad i \in I,$$

где $x^*_{ij}(y_i)$ — оптимальное решение задачи клиентов: найти

$$\min_{x_{ij}} \sum_{j\in J}\sum_{i\in I} g_{ij} x_{ij}$$

при ограничениях:

$$\sum_{i\in I} x_{ij} = 1, \qquad j \in J,$$
$$x_{ij} \le y_i, \qquad i \in I, j \in J,$$
$$x_{ij} \in \{0,1\}, \quad i \in I, j \in J.$$

Напомним, что оптимальные решения z^*_i, y^*_i, $i \in I$, этих задач связаны соотношением $z^*_i = 1 - y^*_i$, $i \in I$, и оптимальные значения целевых функций на этих решениях совпадают.

Заменим условие целочисленности переменных $z_i \in \{0,1\}, i \in I$ на условие их принадлежности отрезку [0,1]. Для полученной задачи выпишем функцию Лагранжа с множителями $\lambda, \mu_i \ge 0, \sigma_i \ge 0, i \in I$:

$$L(z,\lambda,\mu,\sigma) = P(z) + \lambda(m - p - \sum_{i\in I} z_i) + \sum_{i\in I}\sigma_i(z_i - 1) - \sum_{i\in I}\mu_i z_i.$$

Обозначим через $P'_i(z)$ частную производную от функции $P(z)$ по переменной z_i. Тогда условия оптимальности Куна–Такера имеют вид:

$$\frac{\partial L}{\partial z_i}(z,\lambda,\mu,\sigma) = P'_i(z) - \lambda + \sigma_i - \mu_i, \quad i \in I,$$

$$\sum_{i\in I} z_i = m - p,$$
$$0 \le z_i \le 1, \quad i \in I,$$
$$\sigma_i(z_i - 1) = 0, \quad i \in I,$$
$$\mu_i z_i = 0, \quad i \in I.$$

Определение 10 *Вектор* $(z^*, \lambda^*, \mu^*, \sigma^*)$ *называется седловой точкой относительно окрестности Swap, если*

$$L(z^*, \lambda, \mu, \sigma) \overset{(1)}{\le} L(z^*, \lambda^*, \mu^*, \sigma^*) \overset{(2)}{\le} L(z, \lambda^*, \mu^*, \sigma^*)$$

для любых $\lambda, \mu \ge 0, \sigma \ge 0$ *и любого 0-1 вектора* $z \in Swap(z^*)$.

Теорема 26 *Для любого допустимого решения* y^* *задачи BPMP следующие утверждения эквивалентны:*
1) существуют множители Лагранжа $\lambda^*, \mu_i^* \ge 0, \sigma_i^* \ge 0, i \in I$ *такие, что вектор* $(z(y^*), \lambda^*, \mu^*, \sigma^*)$ *является седловой точкой относительно окрестности Swap функции L;*
2) y^* *является локальным минимумом задачи BPMP относительно окрестности Swap;*
3) $z(y^*)$ *удовлетворяет условиям Куна–Такера.*

Доказательство. Проверим, что из 1) следует 2). Пусть $(z(y^*), \lambda^*, \mu^*, \sigma^*)$ седловая точка относительно окрестности *Swap*. Пусть $z^* = z(y^*)$. Из неравенства (1) получаем

$$L(z^*, \lambda^*, \mu^*, \sigma^*) \overset{(3)}{=} \sup_{\lambda,\mu\ge 0,\sigma\ge 0} L(z^*, \lambda, \mu, \sigma) \overset{(4)}{=} P(z^*).$$

Равенство (3) очевидно. Проверим (4). Поскольку $z_i^* - 1 < 0$ или $z_i^* > 0$, то соответствующие множители Лагранжа σ_i^* или μ_i^* равны 0, в противном случае равенство (3) неверно. Следовательно, выполняются условия дополняющей нежесткости

$$\lambda^*(\sum_{i\in I} z_i^* - m + p) = 0, \quad \sigma_i^*(z_i^* - 1) = 0, \quad \mu_i^* z_i^* = 0, i \in I,$$

таким образом, (4) доказано. Из (2), (3) и (4) получаем, что

$$P(z^*) \le L(z, \lambda^*, \mu^*, \sigma^*) \text{ для всех } z \in Swap(z^*).$$

Так как любой вектор $z \in Swap(z^*)$ является допустимым решением задачи, то

$$F(y^*) = P(z^*) \le P(z) + \lambda^*(\sum_{i\in I} z_i - m + p) + \sum_{i\in I} \sigma_i^*(z_i - 1) - \sum_{i\in I} \mu_i^* z_i \le P(z) =$$

$= F(y(z))$, где $y(z) \in Swap(y^*)$.
Следовательно, y^* является локальным минимумом относительно окрестности *Swap*.

Проверим, что из 2) следует 3). Пусть y^* — локальный минимум относительно окрестности *Swap*. Тогда булев вектор $z^* = z(y^*)$ удовлетворяет ограничению $\sum_{i\in I} z_i^* = m - p$ и является локальным минимумом для $P(z)$ по окрестности *Swap*. Докажем, что существуют такие множители $\lambda^*, \mu_i^* \ge 0, \sigma_i^* \ge 0, i \in I$, что вектор $(z^*, \lambda^*, \mu^*, \sigma^*)$ является седловой точкой относительно окрестности *Swap* функции L. Пусть $P''_{i_0 i_1}(z)$ обозначает вторую частную производную функции $P(z)$ по переменным z_{i_0} и z_{i_1}. Введем следующие обозначения

$$\triangle^{i_0}_{-i_1}(y) = P'_{i_0}(z) z_{i_0} - P''_{i_0 i_1}(z) z_{i_0} z_{i_1},$$

$$\triangle^{i_1}_{-i_0}(z) = P'_{i_1}(z) z_{i_1} - P''_{i_0 i_1}(z) z_{i_0} z_{i_1},$$

$$\triangle_{-i_0-i_1}(z) = P(z) - P''_{i_0 i_1}(z) z_{i_0} z_{i_1} - \triangle^{i_0}_{-i_1}(z) - \triangle^{i_1}_{-i_0}(z).$$

Следовательно,

$$P(z) = P''_{i_0 i_1}(z) z_{i_0} z_{i_1} + \triangle^{i_0}_{-i_1}(z) + \triangle^{i_1}_{-i_0}(z) + \triangle_{-i_0-i_1}(z). \tag{5}$$

Возьмём $z \in Swap(z^*)$, $z^*_{i_0} = 0$, $z^*_{i_1} = 1$, $z_{i_0} = 1$, $z_{i_1} = 0$ и $z_i = z^*_i$ для всех $i \ne i_0, i_1$. Учитывая (5), получаем

$$P(z^*) = \triangle^{i_1}_{-i_0}(z^*) + \triangle_{-i_0-i_1}(z^*) = P'_{i_1}(z^*) + \triangle_{-i_0-i_1}(z^*),$$

$$P(z) = \triangle^{i_0}_{-i_1}(z) + \triangle_{-i_0-i_1}(z) = P'_{i_0}(z^*) + \triangle_{-i_0-i_1}(z^*).$$

Итак,

$$P(z^*) - P(z) = P'_{i_1}(z^*) - P'_{i_0}(z^*). \tag{6}$$

Из локальной оптимальности z^* имеем

$$P'_{i_1}(z^*) - P'_{i_0}(z^*) \le 0. \tag{7}$$

Рассмотрим индексы i_0^*, i_1^* такие, что

$$P'_{i_0^*}(z^*) = \min_{i:\ z_i^*=0} P'_i(z^*),\ P'_{i_1^*}(z^*) = \max_{i:\ z_i^*=1} P'_i(z^*).$$

Подставляя i_0^*, i_1^* в (7), получаем $P_{i_1^*}^{'}(z^*) \leq P_{i_0^*}^{'}(z^*)$.
Положим $\lambda^* \in [P_{i_1^*}^{'}(z^*), P_{i_0^*}^{'}(z^*)]$ и

$$\mu_i^* = \begin{cases} P_i^{'}(z^*) - \lambda^* \geq 0, & \text{если } z_i^* = 0, \\ 0 & \text{иначе,} \end{cases}$$

$$\sigma_i^* = \begin{cases} \lambda^* - P_i^{'}(z^*) \geq 0, & \text{если } z_i^* = 1, \\ 0 & \text{иначе.} \end{cases}$$

Получаем $\mu^* \geq 0$, $\sigma^* \geq 0$ и условия дополняющей нежесткости

$$\lambda^*(\sum_{i \in I} z_i^* - m + p) = 0, \quad \sigma_i^*(z_i^* - 1) = 0, \quad \mu_i^* z_i^* = 0, \quad i \in I.$$

Кроме того,

$$\frac{\partial L}{\partial z_i}(z^*, \lambda^*, \mu^*, \sigma^*) = P_i^{'}(z^*) - \lambda^* + \sigma_i^* - \mu_i^* = 0, \ i = 1, \ldots, m.$$

Значит, из 2) следует 3).

Проверим, что из 3) следует 1). Из условий дополняющей нежесткости получаем

$$L(z^*, \lambda^*, \mu^*, \sigma^*) = P(z^*).$$

Возьмём $z \in \mathit{Swap}(z^*)$, $z_{i_0}^* = 0$, $z_{i_1}^* = 1$, $z_{i_0} = 1$, $z_{i_1} = 0$ и $z_i = z_i^*$ для всех $i \neq i_0, i_1$. Тогда

$$L(z, \lambda^*, \mu^*, \sigma^*) = P(z) + \lambda^*(\sum_{i \in I} z_i - m + p) + \sum_{i \in I} \sigma_i^*(z_i - 1) - \sum_{i \in I} \mu_i^* z_i =$$

$$P(z) - \sigma_{i_1}^* - \mu_{i_0}^*.$$

Так как $P_i^{'}(z) - \lambda + \sigma_i - \mu_i = 0, \ i = 1, \ldots, m$, и выполнены условия дополняющей нежесткости, то получаем $\sigma_{i_1}^* = \lambda^* - P_{i_1}^{'}(z^*)$, $\mu_{i_0}^* = P_{i_0}^{'}(z^*) - \lambda^*$. Следовательно,

$$L(z, \lambda^*, \mu^*, \sigma^*) = P(z) + P_{i_1}^{'}(z^*) - P_{i_0}^{'}(z^*).$$

Учитывая (6), получаем

$$L(z^*, \lambda^*, \mu^*, \sigma^*) = P(z^*) = P(z) + P_{i_1}^{'}(z^*) - P_{i_0}^{'}(z^*) = L(z, \lambda^*, \mu^*, \sigma^*).$$

Таким образом, доказали (2). Заметим, что

$$L(z^*, \lambda, \mu, \sigma) = P(z^*) + \lambda(\sum_{i \in I} z_i^* - m + p) + \sum_{i \in I} \sigma_i (z_i^* - 1) - \sum_{i \in I} \mu_i z_i^* =$$

$$= P(z^*) - \sum_{i:\ z_i^*=0} \sigma_i - \sum_{i:\ z_i^*=1} \mu_i \leq P(z^*) = L(z^*, \lambda^*, \mu^*, \sigma^*).$$

□

Заметим, что доказанное утверждение касалось задачи ВРМР и основывалось на эквивалентности этой задачи и задачи минимизации произвольного полинома от булевых переменных. Однако многостадийная задача размещения тоже обладает этим свойством. Следовательно, для этой задачи с дополнительным ограничением на число открываемых наборов предприятий (технологических цепочек) также будет справедливо аналогичное утверждение.

Если удалить требование ограниченности числа открываемых предприятий и заменить окрестность *Swap* на окрестность *Flip*, то можно получить аналогичные утверждения и для простейшей задачи размещения и ее обобщений. Впервые такие результаты были получены А.В.Плясуновым.

4.6 Приближенный локальный поиск

Так как вопрос о совпадении классов PLS и P_{PLS} остается открытым, логично исследовать возможности приближенного поиска локальных оптимумов за полиномиальное время. Пусть $F(s) > 0$, $s \in Sol$. Допустимое решение s^ε задачи (OP, N) называют *ε-локальным минимумом* [178], если $(F(s^\varepsilon) - F(s))/F(s) \leq \varepsilon$ для всех $s \in N(s^\varepsilon)$ и заданного $\varepsilon > 0$. Решение s^ε может не быть локальным минимумом. В его окрестности могут содержаться решения с меньшим значением целевой функции, но их относительное уклонение от s^ε не превышает ε.

Определение 11 *Семейство алгоритмов $(A_\varepsilon)_{\varepsilon>0}$ для задачи (OP, N) называется ε-локальной оптимизационной схемой, если алгоритм A_ε позволяет находить ε-локальный минимум для любого положительного ε. Такое семейство называют полностью полиномиальной ε-локальной оптимизационной схемой, если время работы алгоритма A_ε ограничено полиномом от длины записи исходных данных и величины $1/\varepsilon$.*

Установлено [178], что любая задача локального поиска из класса PLS с линейной целевой функцией имеет полностью полиномиальную ε-локальную оптимизационную схему. Рассмотрим оптимизационную задачу OP, в которой множество допустимых решений Sol является семейством подмножеств конечного множества $E = \{1, \ldots, n\}$. Каждому элементу $e \in E$ приписан вес $f_e > 0$, и целевая функция задачи $F(s) : 2^E \to Q_+$ задается равенством $F(s) = \sum\limits_{e \in s} f_e$.

Алгоритм A_ε для нахождения ε-локального минимума по заданной окрестности N начинает свою работу с произвольного допустимого решения s^0. Согласно определенному правилу весовые коэффициенты f_e, $e \in E$ модифицируются, и для новой задачи применяется стандартный алгоритм локального улучшения. Если в ходе его работы не удается получить решение со значением меньше чем $F(s^0)/2$, то алгоритм останавливается в локальном минимуме для модифицированной задачи и выдает это решение в качестве ответа. В противном случае решение s^0 заменяется новым полученным решением, снова меняются весовые коэффициенты и процесс продолжается. Ниже приводится описание алгоритма. Заметим, что на шаге 2 происходит округление весовых коэффициентов с ростом до ближайшего целого.

Алгоритм ε-Local Search

1. Найти $s^0 \in Sol$ и положить $i = 0$.

2. Вычислить $K = F(s^0), \quad q = \varepsilon K/(2n(1+\varepsilon)), \quad f'_e = \lceil \frac{f_e}{q} \rceil q$, $e \in E$.

3. Положить $j = 0$, $s^{ij} = s^i$.

4. Повторять до тех пор, пока $F(s^{ij}) \le K/2$;
если s^{ij} — локальный минимум, то положить $s^\varepsilon = s^{ij}$ и STOP,
иначе найти лучшее соседнее решение $s^{ij+1} \in N(s^{ij})$, $F(s^{ij+1}) < F(s^{ij})$, и положить $j = j + 1$.

5. Положить $s^{i+1} = s^{ij}$, $i = i + 1$ и вернуться на шаг 2.

Теорема 27 *[178]. Алгоритм ε-Local Search выдает ε-локальный минимум, и его время работы ограничено полиномом от длины записи исходных данных и величины $1/\varepsilon$.*

Рассмотрим подробнее работу алгоритма. Пусть s^ε результат его работы и $s \in N(s^\varepsilon)$. Тогда

$$F(s^\varepsilon) = \sum_{e \in s^\varepsilon} f_c \le \sum_{e \in s^\varepsilon} \lceil \frac{f_e}{q} \rceil q \le \sum_{e \in s} \lceil \frac{f_e}{q} \rceil q \le \sum_{e \in s} q(\frac{f_e}{q} + 1) \le$$

$$\sum_{e \in s} f_e + nq = F(s) + nq.$$

Если $F(s) \ge K/2$, то

$$\frac{F(s^\varepsilon) - F(s)}{F(s)} \le \frac{nq}{F(s)} \le \frac{nq}{F(s^\varepsilon) - nq} \le \frac{2nq}{K - 2nq} = \varepsilon.$$

Оценим время работы алгоритма. Шаг 1 является полиномиальным, так как рассматриваемая задача по предположению принадлежит классу PLS. На шаге 4 происходит локальное улучшение на величину, не меньшую q. Поэтому число итераций локального спуска не превосходит величины $O(n(1+\varepsilon)/\varepsilon) = O(n/\varepsilon)$. Возвращение на шаг 2 происходит не более чем $\log F(s^0)$ раз. Поэтому общее число итераций алгоритма не превышает величины $O(n \log F(s^0)/\varepsilon)$. При более детальном анализе можно получить оценку $O(n^2 \varepsilon^{-1} \log n)$.

Таким образом, с точки зрения аппроксимации при поиске локальных оптимумов нет такого расслоения задач по классам, как при поиске глобального оптимума. Например, для простейшей задачи размещения нахождение приближенного решения s с константной точностью $F(s,x) \le kF(s',x)$ для всех $s' \in Sol(x)$, влечет P=NP. Если же матрица транспортных затрат удовлетворяет неравенству треугольника, то такая константа существует [160], но она не может опуститься ниже порога 1.463, если P $\neq$ NP [132]. Существование PTAS установлено только для частного случая, когда матрица транспортных затрат соответствует Евклидовым расстояниям между точками на плоскости [79]. Если же ищется локальный оптимум, то аналог FPTAS, как мы только что убедились, всегда существует.

Следствие 8 *Для дискретных задач размещения, принадлежащих классу PLS и имеющих линейную целевую функцию, алгоритм ε-Local Search является полностью полиномиальной ε-локальной оптимизационной схемой.*

Заметим, что требование принадлежности классу PLS является существенным в данном утверждении. Многие задачи размещения с ограничениями на мощности открываемых предприятий не принадлежат классу

PLS, если P≠NP. Даже при заданных пунктах размещения предприятий оптимальное распределение клиентов между ними равносильно решению обобщенной задачи о назначениях, которая является NP-трудной [25]. Другими словами, поиск допустимого решения задачи при наличии ограничений на мощности предприятий не является полиномиальной процедурой, и алгоритм ε-Local Search уже на первом шаге сталкивается с проблемой выбора начального решения $s^0 \in Sol$.

Заменим в определении приближенного локального оптимума относительное уклонение на абсолютное. Теперь существование полиномиального алгоритма для нахождения ε-приближенного локального оптимума влечёт равенство PLS=P_{PLS}.

Теорема 28 *[178] Если существует алгоритм, который для любого примера некоторой PLS-полной задачи локального поиска (OP, N) с линейной целевой функцией за полиномиальное время находит такое допустимое решение s^ε, что $F(s^\varepsilon) \leq F(s) + \varepsilon$ для всех $s \in N(s^\varepsilon)$ при заданном $\varepsilon > 0$, то PLS=P_{PLS}.*

Чтобы лучше понять это принципиальное отличие, предположим, что целевая функция принимает только целые значения. Для каждого примера x задачи OP построим новый пример x' с тем же множеством допустимых решений $Sol(x) = Sol(x')$ и новой целевой функцией $F(s) = \sum\limits_{e \in s} f'_e$, где $f'_e = f_e(1 + \varepsilon), e \in E$. Применим алгоритм к этому примеру, и пусть s' — результат его работы. Тогда $F'(s') - F'(s) \leq \varepsilon$ для всех $s \in N(s')$. Следовательно, $F(s') - F(s) \leq \varepsilon/(\varepsilon + 1) < 1$ для всех $s \in N(s')$ и s' — локальный оптимум для x. Таким образом, для дискретных задач размещения замена относительного уклонения на абсолютное в определении приближенного локального оптимума ведет к скачку в вычислительной сложности нахождения таких решений. Если же для дискретных задач размещения, принадлежащих классу PLS и имеющих линейную целевую функцию, найдется такая полностью полиномиальная ε-локальная оптимизационная схема $(A_\varepsilon)_{\varepsilon>0}$, что время работы A_ε зависит полиномиально от длины входа и $\log 1/\varepsilon$, то PLS = P_{PLS} [178].

4.7 Погрешность локальных оптимумов

Напомним, что окрестность называют *точной*, если любой локальный минимум является глобальным. Существование точной полиномиально

проверяемой окрестности делает локальный спуск точным методом, как, например, симплекс–метод. Однако для задачи коммивояжера существование такой окрестности влечет P=NP [50]. В данном разделе будет показана бесперспективность поиска таких окрестностей также для задачи о p-медиане и ее обобщений. Кроме того, будет установлено, что локальный минимум для любой полиномиально проверяемой окрестности может быть больше глобального в произвольное число раз, а количество локальных минимумов может расти экспоненциально с ростом размерности задачи. В конце раздела будет приведено семейство исходных данных, на котором расстояние от глобального минимума до ближайшего локального минимума равно $2p$, т. е. равно диаметру допустимой области. Более того, любой путь от локального минимума в глобальный проходит через область больших значений целевой функции, что является серьезным препятствием для таких методов как имитация отжига, поиск с запретами и др.

Говоря о погрешности приближенного решения s для входа x оптимизационной задачи на минимум, будем иметь в виду отношение $F(x,s)/Opt(x)$, где $Opt(x)$ — оптимальное значение целевой функции. Известно [122], что для любого целого $\rho > 1$ задача вычисления приближённого решения с гарантированной оценкой точности ρ для задачи о p-медиане является NP-трудной. Этот результат основывается на полиномиальной сводимости задачи о вершинном покрытии к задаче о p-медиане.

Задача о вершинном покрытии формулируется следующим образом [25]. Задан граф $G = (V, E)$ и целое положительное число k. Требуется узнать, существует ли подмножество вершин $V^{'} \subset V$ мощности не более k такое, что любое ребро из множества E инцидентно хотя бы одной вершине из множества $V^{'}$? Множество $V^{'}$ называют вершинным покрытием.

Рассмотрим пример задачи о p-медиане со следующими входными данными: $I = V, J = E, p = k$, и

$$c_{ij} = \begin{cases} 1, & \text{если вершина } i \text{ инцидентна ребру } e_j, \\ \rho(|E|+1) & \text{в противном случае.} \end{cases}$$

Исходный граф G содержит вершинное покрытие размера k тогда и только тогда, когда оптимальное значение данной задачи о p-медиане равно

величине $|E|$. Если граф G не содержит вершинное покрытие размера k, то значение целевой функции на произвольном допустимом решении можно оценить снизу величиной

$$|E| - 1 + \rho(|E| + 1) = \rho|E| + |E| + \rho - 1 > \rho|E|.$$

Таким образом, если для задачи о p-медиане существует приближенный алгоритм с гарантированной оценкой ρ, то с его помощью можно дать точный ответ в задаче о вершинном покрытии.

Предположим теперь, что для некоторой полиномиально проверяемой окрестности существует такая константа ρ, что все локальные минимумы в задаче о p-медиане превышают глобальный минимум не более чем в ρ раз. Тогда алгоритм локального спуска будет всегда давать ρ-приближенные решения. В частности, такие решения он получит и для указанного семейства исходных данных. Но для этого семейства алгоритм требует не более $|E|$ шагов для достижения локального минимума. Действительно, на каждом шаге локального спуска происходит уменьшение целевой функции на величину, кратную $\rho(|E| + 1) - 1$. Другими словами, алгоритм локального улучшения является полиномиальным для этого семейства исходных данных. Таким образом, получаем точный полиномиальный алгоритм для решения задачи о вершинном покрытии. Значит, предположение о существовании окрестности с указанными свойствами влечет P=NP.

Теорема 29 *Если P≠NP, то для любого $\rho > 1$ и любой полиномиально проверяемой окрестности найдутся исходные данные задачи о p-медиане, для которых существует локальный минимум, отклоняющийся более чем в ρ раз от оптимального значения задачи.*

При $\rho = 1$ получаем следующее утверждение.

Следствие 9 *Если P ≠NP, то для задачи о p-медиане не существует точных полиномиально проверяемых окрестностей.*

Заметим, что все приведенные выше рассуждения сохраняют свою силу при замене константы ρ на экспоненциальную функцию $2^{q(n,m)}$, где $q(n, m)$ — произвольный полином от размерности задачи.

В [212] для произвольной задачи (OP, N) из класса PLS анонсированы без доказательства два важных утверждения, которые позволяют оценивать качество локальных оптимумов как приближенных решений

оптимизационной задачи OP. При формулировке первого из них предполагалась полиномиальная ограниченность значений целевой функции. Покажем, что этот результат верен и в более общей ситуации. Для этого введем понятие псевдо-полиномиальной ограниченности, которое означает, что диаметр области значений целевой функции растет не быстрее, чем значение некоторого полинома от длины входа $|x|$ и величины $Max(x)$, которая означает наибольшее по величине число из входа задачи x [25, 80]. Приведем точную формулировку данного определения.

Определение 12 *Задача OP называется псевдо-полиномиально ограниченной, если существует полином q от переменных $|x|$ и $Max(x)$ такой, что для любого входа x и любого решения $s \in Sol(x)$ выполняется неравенство*

$$F(s,x) - Opt(x) \leq q(|x|, Max(x)).$$

Множество оптимизационных задач, удовлетворяющих данному определению, обозначим через NPO_{PPB}.

Теорема 30 *Пусть $OP \in NPO_{PPB}$, $(OP, N) \in PLS$ и целевая функция принимает только целочисленные значения. Если $P \neq NP$ и задача вычисления приближённого решения с гарантированной оценкой точности ρ является NP-трудной в сильном смысле, то найдутся исходные данные, для которых существует локальный оптимум, отклоняющийся более чем в ρ раз от оптимального значения.*

Доказательство. Обозначим через Π^{ρ} задачу вычисления приближённого решения с гарантированной оценкой точности ρ для задачи OP. По предположению теоремы она является NP-трудной в сильном смысле. Тогда по определению [25, 80] существует полином q' такой, что подзадача $\Pi^{\rho}_{q'}$, образованная входами x, для которых выполняется неравенство $Max(x) \leq q'(|x|)$, является NP-трудной.

Предположим также, что существует полиномиально проверяемая окрестность N такая, что все локальные оптимумы являются ρ-приближенными решениями. Покажем, что в этом случае подзадача $\Pi^{\rho}_{q'}$ является полиномиально разрешимой, и получим (см. [25], теорема 7.2) P = NP.

Возьмём произвольный вход x этой подзадачи и некоторое начальное решение s^0, которое существует по определению класса PLS. Из псевдополиномиальной ограниченности задачи OP получаем

$$F(s^0, x) - Opt(x) \leq q(|x|, Max(x)) \leq q(|x|, q'(|x|)).$$

Так как целевая функция по условиям теоремы принимает целые значения, то из последнего неравенства следует, что алгоритм локального спуска, начиная с решения s^0 за полиномиальное время найдет некоторый локальный оптимум. Таким образом, если все локальные оптимумы являются ρ-приближенными решениями, то получим, что NP-трудная задача $\Pi^{\rho}_{q'}$ является полиномиально разрешимой, что противоречит предположению P$\neq$NP. Следовательно, найдутся исходные данные задачи OP, для которых существует локальный оптимум, отклоняющийся более чем в ρ раз от оптимального значения. □

Следствие 10 *Если $P \neq NP$ и задача $OP \in NPO_{PPB}$ является NP-трудной в сильном смысле, то для неё не существует точных полиномиально проверяемых окрестностей.*

Предыдущие результаты можно усилить, рассмотрев более широкий класс задач локального поиска, для которых диаметр графа переходов ограничен полиномом от $|x|$ и $Max(x)$. Теорема и ее следствие будут выполняться для всех задач из этого класса. Легко проверить, что задачи из класса NPO_{PPB} попадают в этот более широкий класс. Обратное, вообще говоря, неверно.

Теорема 31 *Если NP $\neq$co-NP и задача вычисления приближённого решения с гарантированной оценкой точности ρ для задачи OP является NP-трудной, то найдутся исходные данные, для которых существует локальный оптимум в задаче $(OP, N) \in PLS$, отклоняющийся более чем в ρ раз от оптимального значения.*

Доказательство. По предположению теоремы задача Π^{ρ} является NP-трудной. Следовательно, существует NP-полный язык π, который сводится по Тьюрингу к задаче Π^{ρ} за полиномиальное время. Это означает, что существует машина Тьюринга T с оракулом [25, 181], которая за полиномиальное время распознает язык π. В процессе работы машина обращается к оракулу, который, получив вход задачи Π^{ρ}, выдает ρ-приближенное решение. По определению оракульной машины [25] такое обращение принимается за один шаг её работы. Получив приближенное решение, машина продолжает работу в соответствии со своими инструкциями. Будем считать, что существует полином p такой, что T выполняет не более $p(|x|)$ шагов на каждом входе x [181].

Предположим, что существует полиномиально проверяемая окрестность N такая, что все локальные оптимумы являются ρ-приближенными решениями. Покажем, что в этом случае язык π^c, являющийся дополнением языка π, принадлежит NP. Тогда (см. [25], теорема 7.2) NP = co-NP.

По машине T построим новую машину T', которая для произвольного входа x будет определять принадлежность этого входа языку π^c за полиномиальное недетерминированное время. Новая машина отличается от старой тем, что обращение к оракулу она заменяет отгадыванием локального оптимума в задаче (OP, N). Это можно сделать за полиномиальное недетерминированное время, так как по определению класса PLS длина записи локального оптимума и проверка его локальной оптимальности осуществляются за полиномиальное время. По предположению этот локальный оптимум является ρ-приближенным решением. Значит, новая машина будет давать тот же ответ, что и старая. Так как число шагов машины T не превышает $p(|x|)$, то количество предъявлений локального оптимума тоже ограничено полиномом $p(|x|)$. Таким образом, новая машина T' работает полиномиальное недетерминированное время. Если на последнем шаге работы машины T' получим ответ "да", то $x \notin \pi^c$, иначе $x \in \pi^c$. Итак, язык π является NP-полным, его дополнение π^c лежит в NP, значит NP=co-NP. □

Следствие 11 *Если NP ≠co-NP и задача OP является NP-трудной, то для неё не существует точных полиномиально проверяемых окрестностей.*

В заключение данной главы приведём семейство исходных данных задачи о p-медиане, в котором для любой из окрестностей $Swap, LK$, LK_1, FM, FM_1, k-$Swap$ существует экспоненциальное число локальных минимумов, ни один из которых не является глобальным. Более того, все они в ρ раз отличаются от глобального оптимума. Матрица c_{ij} в этом примере имеет p блоков размерности $p \times p$ и p дополнительных строк, которые соответствуют оптимальному решению задачи. Все элементы каждого блока равны ρ. Таким образом, у матрицы имеется p^2 столбцов и $p^2 + p$ строк (см. рис. 18).

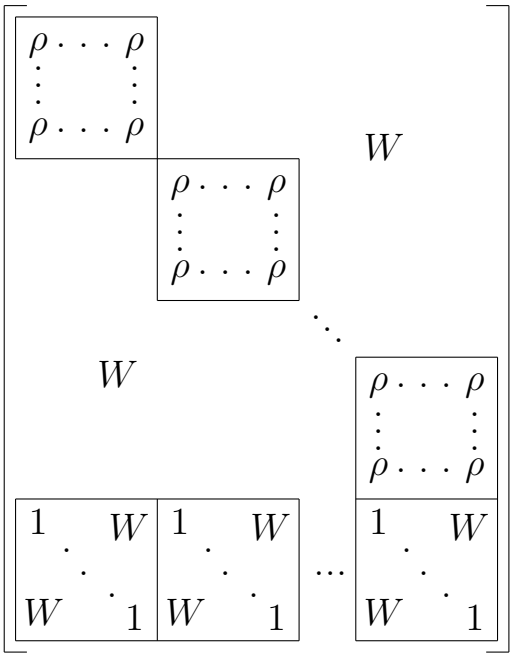


Рис. 18. *Матрица* c_{ij}

Пусть $W > \rho p^2$. В первых p^2 строках элементы, которые не попали в блоки, положим равными W. В последних p строках разместим p квадратных матриц размера $p \times p$, в которых по диагонали стоят единицы, а остальные элементы равны W. Легко проверить, что последние p строк дают единственный глобальный минимум. Его значение равно p^2. Кроме него имеется p^p допустимых решений со значением целевой функции ρp^2. Каждое из них содержит ровно одну строку из каждого блока. На всех оставшихся решениях значение целевой функции строго больше ρp^2. Окрестности $FM_1, FM, Swap, LK_1, LK, k-Swap$ порождают p^p локальных минимумов, не являющихся глобальными. Это нестрогие локальные минимумы с одним и тем же значением целевой функции. Окрестность каждого из них содержит другие локальные минимумы. Они образуют своего рода "плато"экспоненциальной мощности. Попадая на него, такие методы как поиск с запретами, имитация отжига и др. теряют способность к целенаправленному поиску. Фактически уйти с такого плато метод уже не может. Блуждание по плато является аналогом зацикливания, которое наблюдается в симплекс-методе при попадании в вырожденную вершину. Для преодоления таких ловушек требуются специальные процедуры. Рассмотрим граф соседства [212], в котором вершинами являются допустимые решения, а дуга ставится в том случае, если одно решение принадлежит окрестности другого. Для данного семейства ис-

ходных данных любой путь в таком графе от локального минимума в глобальный будет проходить через решения с высокими значениями целевой функции. Для окрестности $Swap$ потребуется как минимум $\lfloor p/2 \rfloor$ шагов с ростом целевой функции вдоль любого такого пути. Образно говоря, глобальный минимум отделен от всех локальных минимумов своего рода "горным хребтом", преодоление которого может оказаться трудным для методов локального поиска. Увеличение константы W приводит к падению вероятности преодоления этого препятствия пороговыми методами, например, методом имитации отжига. Рост параметра p приводит к увеличению кратчайшего пути от локального оптимума к глобальному, что отрицательно влияет на эффективность поиска с запретами. Генетические алгоритмы, имея начальную популяцию из первых p^2 строк, сталкиваются с теми же трудностями, т. к. оператор скрещивания не может дать последние p строк, и только случайные мутации могут это сделать. Если же генетический алгоритм использует процедуру локального спуска на стадии улучшения "потомства", то влияние мутаций будет сильно ослаблено и вероятность получить глобальный оптимум окажется еще ниже. Таким образом, данное семейство примеров, несмотря на свою простую структуру, является сложным для большинства метаэвристик.

Глава 5

Равновесия по Нэшу в игровых моделях размещения

Многие оптимизационные задачи можно переформулировать в виде игры. В области дискретных задач размещения в последнее время активно исследуются игры Штакельберга. Игроки в таких играх неравноправны. Лидер делает ход первым, и его решение определяет систему ограничений для других игроков. Интерес к таким моделям связан как с многочисленными приложениями, например, с исследованием иерархических систем управления, так и с вычислительной сложностью таких задач. В настоящей главе исследуется другой класс игр. Все игроки равноправны, не образуют коалиций и преследуют чисто эгоистические интересы. Они одновременно принимают решения, стараясь максимизировать собственную прибыль от открытия предприятий и обслуживания клиентов. Тот факт, что игроки не образуют коалиций, приводит к конкуренции в области ценообразования, падению цен на рынке и перераспределению доходов между игроками и потребителями. Решение называют равновесным, если ни один из игроков не может увеличить свою прибыль при условии, что другие игроки не меняют свои решения. Понятие равновесного решения в чистых стратегиях оказывается тесно связанным с понятием локального оптимума в соответствующей оптимизационной задаче. Фактически, удается установить взаимно-однозначное соответствие между равновесными по Нэшу решениями и локальными оптимумами. Таким образом, определяется сложностной статус задач поиска равновесных решений. В частности, для данной игры поиск равновесия не может быть NP-полной задачей, если NP$\neq$co-NP. Показано, что задача поиска равновесия принадлежит классу PLS и является плотно полной в нем. Отсюда, в частности, следует, что доказательство существования полиномиального алгоритма вычисления равновесного решения было бы

слишком сильным результатом. Все задачи из класса PLS решались бы в этом случае за полиномиальное время. Доказательство обратного утверждения приводит к выводу, что P≠NP. Таким образом, вопрос о сложности нахождения равновесных решений в чистых стратегиях является интригующим и может иметь грандиозные последствия.

5.1 Игровая модель размещения предприятий

Рассмотрим ситуацию, когда p фирм одновременно открывают свои предприятия для обслуживания клиентов. Цель каждой фирмы (игрока) состоит в максимизации прибыли. Будем предполагать, что игроки принимают решения независимо друг от друга, не образуют коалиций и преследуют исключительно эгоистические интересы. Для простоты изложения будем считать, что каждый игрок может открыть не более одного предприятия. Возможные места их размещения зададим множеством I. Несколько игроков могут открывать свои предприятия в одном и том же месте. Затраты на открытие предприятий k-м игроком зададим величинами $f_i^k \geq 0$, $i \in I$.

Пусть величина r_j задает максимальную цену, которую клиент j согласен заплатить за предлагаемую продукцию, и матрица (c_{ij}^k) показывает затраты на обслуживание клиентов из открываемых предприятий. Без ограничения общности можно считать, что $c_{ij}^k \leq r_j, j \in J$. В противном случае положим $c_{ij}^k = r_j$ и тогда игрок не будет иметь прибыль от обслуживания этого клиента.

Предположим, что игроки открыли предприятия $i_1, i_2, \ldots, i_p$. Какую цену q_j заплатит j-й клиент за их продукцию? Если бы игроки объединились, стремясь получить максимальную прибыль, то они установили бы максимально доступную цену на свою продукцию, т.е. $q_j = r_j$, $j \in J$, и весь доход присвоили себе. Что происходит, когда игроки действуют независимо друг от друга? Обозначим через c_j минимальные производственно-транспортные затраты по обслуживанию j-го клиента из открываемых предприятий:

$$c_j = \min\{c_{i_1 j}^1, c_{i_2 j}^2, \ldots, c_{i_p j}^p\}, \quad j \in J.$$

Цена q_j не может быть меньше c_j, так как ни одному из игроков не выгодно поставлять продукцию по столь низкой цене. Предположим, что игроки выставили цену $q_j = r_j$. Клиенту все равно, какой из поставщиков будет его обслуживать. Так как каждый из игроков хочет оказаться

поставщиком, то они начнут снижать цену. Пусть

$$i(j) = \arg\min\{c^1_{i_1 j}, c^2_{i_2 j}, \ldots, c^p_{i_p j}\}$$

и c'_j — второй по величине минимальный элемент среди $c^1_{i_1 j}, c^2_{i_2 j}, \ldots, c^p_{i_p j}$. Процесс падения цены остановится на c'_j, когда игрок $i(j)$ останется единственным поставщиком, кому все еще выгодно обслуживание j-го клиента. Ниже опускать цену нецелесообразно. Выше поднимать цену нельзя, так как появляется конкурент, которому также становится выгодным обслуживание этого клиента. Получаем $q_j = c'_j$, $j \in J$. Если $c_j = c'_j$ для некоторого $j \in J$, то $q_j = c_j$, и обслуживание этого клиента не приносит прибыли ни одному из игроков.

Когда игроки действовали сообща, они держали цену q_j равной r_j и присваивали весь доход. Теперь $q_j = c'_j$ и прибыль $r_j - c_j$ делится между поставщиком $i(j)$, который получает $q_j - c_j$, и клиентом, который экономит величину $r_j - q_j$.

Обозначим через Γ_k множество клиентов, обслуживаемых k-м игроком при заданном выборе $i_1, \ldots, i_p$. Тогда прибыль k-го игрока равна

$$w_k = \sum_{j \in \Gamma_k} (q_j - c_j) - f^k_{i_k},$$

суммарная экономия клиентов определяется величиной

$$\nu(i_1, \ldots, i_p) = \sum_{j \in J} (r_j - q_j),$$

суммарная прибыль игроков и экономия клиентов составляет величину

$$\mu(i_1, \ldots, i_p) = \sum_{j \in J} (r_j - c_j) = \sum_{k=1}^{p} w_k + \nu(i_1, \ldots, i_p).$$

Решение $(i_1, \ldots, i_p)$ называют равновесием по Нэшу, или равновесным решением, если ни один из игроков не может увеличить свою прибыль при условии, что другие игроки не меняют свой выбор. Оптимальным решением называют такое решение, при котором величина $\mu(i_1, \ldots, i_p)$ достигает максимального значения. Такое решение действительно является наилучшим для общества, так как приносит наибольший «эффект» от размещения производства и выпуска продукции. Ниже будет показано, что оптимальное решение является равновесным, но не каждое равновесное решение является оптимальным. Для таких игр в [161] вводится

понятие цены анархии. Эта величина равна отношению оптимального решения к наихудшему среди равновесий по Нэшу. Известно [212], что для данной игры цена анархии не превосходит 2. Однако вопрос о трудоемкости нахождения равновесных решений до сих пор остается открытым. Неясно, можно ли найти хоть одно равновесное решение за полиномиальное время от длины записи исходных данных.

5.2 Связь с локальными оптимумами

Представим задачу максимизации величины $\mu(i_1, \ldots, i_p)$ как задачу целочисленного линейного программирования. Введем следующие переменные:

$$x_i^k = \begin{cases} 1, & \text{если игрок } k \text{ открывает предприятие в пункте } i, \\ 0 & \text{в противном случае,} \end{cases}$$

$$y_{ij}^k = \begin{cases} 1, & \text{если клиент } j \text{ обслуживается игроком } k \text{ из предприятия } i, \\ 0 & \text{в противном случае.} \end{cases}$$

Тогда задача поиска максимальной суммарной прибыли игроков и экономии клиентов может быть сформулирована следующим образом:

$$\max \sum_{j \in J} \sum_{k=1}^{p} \sum_{i \in I} \left(r_j - c_{ij}^k y_{ij}^k\right) - \sum_{k=1}^{p} \sum_{i \in I} f_i^k x_i^k$$

при условиях:

$$\sum_{k=1}^{p} \sum_{i \in I} y_{ij}^k \le 1, \quad j \in J,$$

$$y_{ij}^k \le x_i^k, \quad i \in I, j \in J, k = 1, \ldots, p,$$

$$\sum_{i \in I} x_i^k \le 1; \quad k = 1, \ldots, p,$$

$$x_i^k, y_{ij}^k \in \{0, 1\}, \quad i \in I, j \in J, k = 1, \ldots, p.$$

Целевая функция задачи задает суммарный экономический эффект от открытия предприятий и обслуживания клиентов, т.е. величину $\mu(i_1, \ldots, i_p)$. Первое ограничение позволяет каждому клиенту иметь не более одного поставщика. Второе ограничение разрешает обслуживать

клиентов только из открытых предприятий. Последнее ограничение позволяет каждому игроку открывать не более одного предприятия.

Заметим, что в этой задаче несколько игроков могут открывать предприятия в одном месте. Поэтому такая постановка близка по смыслу к задаче о p-медиане и простейшей задаче размещения [40], но не идентична им. Будем называть ее игровой задачей размещения и обозначать через FLG (*Facility Location Game*).

Пусть (x_i^k, y_{ij}^k) — допустимое решение этой задачи, и при данных значениях x_i^k величины y_{ij}^k задают оптимальное распределение клиентов между открытыми предприятиями. Другими словами, будем предполагать, что значения переменных y_{ij}^k определяются по значениям переменных x_i^k и решение можно задавать только этими переменными. Положим $\overline{c}_{ij}^k = r_j - c_{ij}^k$ для всех k, i, j. Тогда задачу можно переписать следующим образом:

$$\max\Big(\sum_{j\in J}\max_{k=1,\dots,p}\max_{i\in I}\overline{c}_{ij}^k x_i^k - \sum_{k=1}^{p}\sum_{i\in I} f_i^k x_i^k\Big)$$

при условиях:

$$\sum_{i\in I} x_i^k \le 1, \quad k = 1,\dots,p,$$

$$x_i^k \in \{0,1\}, \quad i \in I, k = 1,\dots,p.$$

Пусть (x_i^k) — допустимое решение задачи. Под окрестностью $N_1(x_i^k)$ будем понимать множество допустимых решений задачи FLG, которые могут быть получены из решения (x_i^k) выбором некоторого игрока k и заменой его стратегии любой другой допустимой стратегией т.е. закрытием его предприятия, если он его открывал; открытием какого-то одного предприятия, если не было открыто ни одного; заменой уже открытого предприятия на любое другое. Локальным максимумом для данной окрестности будет любое решение, в окрестности которого нет решений с бóльшим значением целевой функции. Введенная окрестность включает в себя целиком окрестность *Swap* и является частью объединения окрестностей *Swap* и *Flip*. Она имеет полиномиальную мощность. Следовательно, задача локального поиска (FLG, N_1) принадлежит классу PLS. Следующая теорема устанавливает тесную связь между локальными максимумами задачи (FLG, N_1) и равновесиями по Нэшу.

Теорема 32 *Между локальными максимумами задачи* (FLG, N_1) *и рав-*

новесными решениями в чистых стратегиях существует взаимно–однозначное соответствие.

Доказательство. Пусть игроки открыли множество предприятий $(i_1, \ldots, i_p)$. Поставим ему в соответствие решение (x_i^k), определяемое следующим образом: $x_i^k = 1$, если $i = i_k$, и $x_i^k = 0$ в противном случае. Убедимся в том, что это решение является локальным максимумом тогда и только тогда, когда $(i_1, \ldots, i_p)$ — равновесное решение. Для этого достаточно показать, что при смене стратегии k-м игроком его прибыль w_k меняется ровно на столько же, насколько меняется целевая функция задачи FLG.

Предположим, что k-й игрок закрывает свое предприятие i_k. Тогда каждый клиент из множества Γ_k потеряет своего поставщика. Новый поставщик — это предприятие другого игрока, которое имело наименьшую стоимость обслуживания c'_j без учета игрока k. Теперь j-й клиент будет обслуживаться именно по этой цене c'_j и значение $\mu(i_1, \ldots, i_p)$ меняется на величину $\sum\limits_{j \in \Gamma_k} (c'_j - c_{i_k j}) - f_{i_k}^k$. Ровно на столько же меняется значение целевой функции задачи FLG при переходе к соседнему решению

$$\overline{x}_i^\tau = \begin{cases} 0, & \text{если } \tau = k, \\ x^\tau, & \text{если } \tau \neq k, \end{cases} \quad i \in I.$$

Аналогичные рассуждения справедливы и при открытии игроком нового предприятия. Если же игрок передвигает свое предприятие на новое место, то такую стратегию можно представить как двухшаговую: закрыть одно предприятие — открыть другое; и снова получаем требуемое. □

Предположим, что игроки могут открывать не одно, а несколько предприятий, скажем, $a_k \geq 1, k = 1, \ldots, p$. Тогда решение будет называться равновесием по Нэшу, если ни один из игроков не сможет увеличить свою прибыль, меняя расположение своих предприятий, предполагая, что другие игроки не меняют своих стратегий. Математическая постановка задачи ЦЛП будет немного другой, точнее, меняется только последнее ограничение. Теперь оно принимает вид

$$\sum_{i \in I} x_i^k \leq a_k, \quad k = 1, \ldots, p.$$

Вместо окрестности N_1 следует рассматривать окрестность N_a, в которой содержатся все решения, полученные из данного выбором игрока и заме-

ной его стратегии на любую другую. Утверждение теоремы по-прежнему сохраняет силу, и доказательство этого утверждения проводится аналогично.

5.3 Сложность нахождения равновесных решений

Основной результат данного раздела состоит в доказательстве плотной PLS–полноты задачи нахождения равновесий по Нэшу в чистых стратегиях. Сначала показана плотная PLS–полнота задачи (FLG, N_1). Затем этот результат обобщен на случай (FLG, N_a), откуда и следует требуемое.

Известно [212], что следующая задача о максимальном разрезе с окрестностью *Flip* является плотно PLS–полной. Задан неориентированный граф $G = (V, E)$ с положительными весами на ребрах $w_e, e \in E$. Допустимым решением задачи является разбиение множества вершин V на два подмножества V_1 и V_2 не обязательно равной мощности. Целевая функция задачи определяется как сумма весов ребер, каждое из которых имеет одну вершину в множестве V_1, а другую вершину в множестве V_2 (суммарный вес разреза). В задаче требуется найти разбиение (V_1, V_2) с максимальным значением целевой функции. Известно, что задача на минимум (*Max-Cut*) является полиномиально разрешимой, в то время как задача на максимум (*Max-Cut*) является NP–трудной. Окрестность *Flip* для данного разбиения имеет мощность $|V|$ и состоит из всех разбиений, полученных из данного выбором одной вершины и переносом ее в другое подмножество.

Теорема 33 *Задача (Max-Cut, Flip) плотно PLS–сводится к задаче* (FLG, N_1).

Доказательство. По заданному графу построим исходные данные задачи FLG. Положим $I = \{1, \ldots, 2|V|\}$, $J = \{1, ..., |V| + 2|E|\}$, $p = |V|$, $w^* = \max_{e \in E} |w_e| + 1$, $W = \sum_{e \in E} |w_e| + w^* d$, где величина d является максимальной степенью вершин графа. Положим $f_i^k = 0$ для всех k и i. Сопоставим каждому игроку одну и ту же матрицу c_{ij} следующим образом.

Каждой вершине $v \in V$ поставим в соответствие две строки i_v, i'_v и один столбец j_v, а каждому ребру $e = (j_1, j_2) \in E$ — два столбца

$j_1(e), j_2(e) \in J$. При $j_v = 1, \ldots, |V|$ положим

$$c_{ij_v} = \begin{cases} W, & \text{если } (i = i_v) \vee (i = i'_v), \\ 0 & \text{в противном случае.} \end{cases}$$

Для $e = (j_1, j_2) \in E$ положим

$$c_{ij_1(e)} = \begin{cases} w^* + w_e/2, & \text{если } (i = j_1(e)) \vee (i = j_2(e)), \\ w^* - w_e/2, & \text{если } (i = j'_1(e)) \vee (i = j'_2(e)), \\ 0 & \text{в противном случае,} \end{cases}$$

$$c_{ij_2(e)} = \begin{cases} w^* - w_e/2, & \text{если } (i = j_1(e)) \vee (i = j_2(e)), \\ w^* + w_e/2, & \text{если } (i = j'_1(e)) \vee (i = j'_2(e)), \\ 0 & \text{в противном случае.} \end{cases}$$

Структура матрица (c_{ij}) изображена на рис. 19.

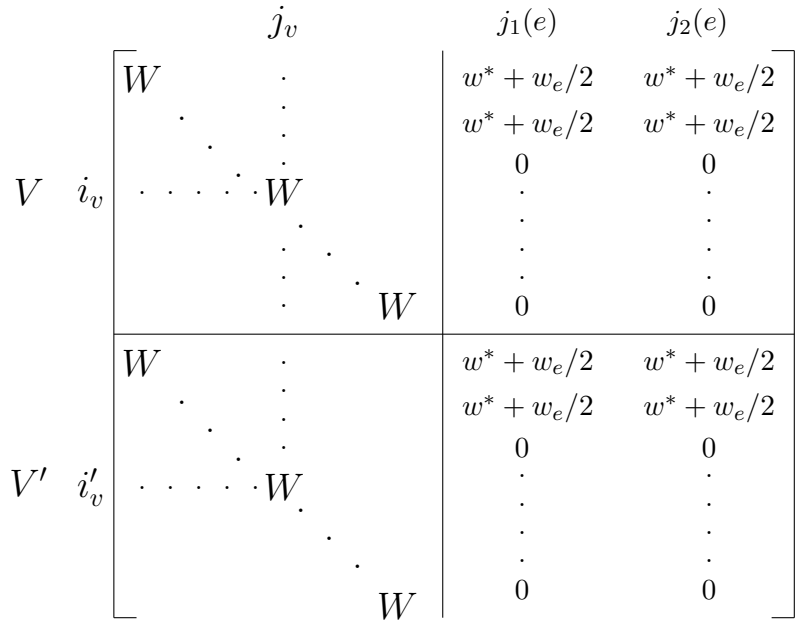


Рис. 19 *Матрица* (c_{ij})

Предложенная конструкция определяет полиномиально вычислимую функцию h из определения PLS-сводимости (см. определение 8). Функцию g зададим следующим образом. Пусть (x_i^k) — допустимое решение задачи FMG. Тогда значением функции g на этом решении является

разрез (V_1^x, V_2^x), где множество V_1^x состоит из вершин v для каждой из которых $x_{i_v}^k = 1$ для подходящего k, а множество V_2^x является его дополнением:

$$V_1^x = \{v \in V \mid \sum_{k=1}^{p} x_{i_v}^k \geq 1\}, \qquad V_2^x = V \setminus V_1^x.$$

Проверим, что разрез (V_1^x, V_2^x) является локальным максимумом относительно окрестности *Flip*, если и только если (x_i^k) — локальный максимум для окрестности N_1. Заметим, что любой локальный максимум (x_i^k) обладает следующим свойством:

для любого $v \in V$ существует единственный номер k такой, что

$$x_{i_v}^k + x_{i'_v}^k = 1.$$

Покажем сначала, что не существует вершины $v \in V$ такой, что $x_{i_v}^{k_1} = x_{i_v}^{k_2} = 1$, при любых $k_1 \neq k_2$. Допустим противное, т.е. что для некоторой вершины $v \in V$ найдутся такие номера. Так как $p = |V|$, то найдется такая вершина $v_0 \in V$, что $x_{i_{v_0}}^k = x_{i'_{v_0}}^k = 0$ для всех k. Построим новое решение $\overline{x}$, которое отличается от прежнего только компонентами (k_1, i_{v_0}) и (k_1, i_v). Положим $\overline{x}_{i_{v_0}}^{k_1} = 1$ и $\overline{x}_{i_v}^{k_1} = 0$. Это решение — соседнее для исходного, и оно имеет большее значение целевой функции. Действительно, так как $x_{i_{v_0}}^{k_1} = x_{i'_{v_0}}^{k_1} = 0$, то слагаемое с номером j_{v_0} в целевой функции задачи FLG равно 0. Полагая $\overline{x}_{i_{v_0}}^{k_1} = 1$, получаем увеличение целевой функции на W. Замена $x_{i_v}^{k_1} = 1$ на $\overline{x}_{i_v}^{k_1} = 0$ не может привести к уменьшению целевой функции, так как $x_{i_v}^{k_2} = 1$. Более того, в целевой функции появляются дополнительные положительные слагаемые $(w^* \pm w/2) > 0$, соответствующие ребрам, инцидентным вершине v_0. Получили противоречие с предположением о том, что (x_i^k) — локальный максимум. Случаи $x_{i_{v'}}^{k_1} = x_{i_v}^{k_2} = 1$ и $x_{i_{v'}}^{k_1} = x_{i_{v'}}^{k_2} = 1$ рассматриваются аналогично.

Итак, если (x_i^k) — локальный максимум в окрестности N_1, то для каждой вершины $v \in V$ найдется единственный номер $k(v)$ такой, что $x_{i_v}^{k(v)} + x_{i_{v'}}^{k(v)} = 1$. Это свойство гарантирует наличие у каждого игрока ровно одной вершины. Для определенности можно считать, что первый игрок получил первую вершину, второй игрок — вторую и т.д. Покажем теперь, что разрез (V_1^x, V_2^x) является локальным максимумом для окрестности *Flip*.

Рассмотрим ребро $e = (j_1(e), j_2(e)) \in E$. Найдем строки i_1, i_2 матрицы (c_{ij}^k), соответствующие вершинам $j_1(e), j_2(e)$. Положим $k_1 = k(j_1(e))$ и

$k_2 = k(j_2(e))$.

Пусть вершины $j_1(e), j_2(e)$ лежат в множестве V_1^x. Тогда $x_{i_1}^{k_1} = x_{i_2}^{k_2} = 1$ и $x_{i'_1}^{k_1} = x_{i'_2}^{k_2} = 0$. Следовательно, в целевой функции слагаемое для $j = j_1(e)$ равно $w^* + w_e/2$, а для $j = j_2(e)$ равно $w^* - w_e/2$, что в сумме даёт величину $2w^*$. Т.е. вес w_e ребра не учитывается как в задаче о максимальном разрезе, так и в задаче FLG.

Рассмотрим случай, когда вершины $j_1(e), j_2(e)$ лежат в множестве V_2^x. Тогда $x_{i_1}^{k_1} = x_{i_2}^{k_2} = 0$ и $x_{i'_1}^{k_1} = x_{i'_2}^{k_2} = 1$. Следовательно, в целевой функции слагаемое для $j = j_1(e)$ оно равно $w^* - w_e/2$, а для $j = j_2(e)$ равно $w^* + w_e/2$. В сумме опять получаем $2w^*$, т.е. вес w_e не учитывается как в задаче о максимальном разрезе, так и в задаче FLG.

Рассмотрим теперь ребро, концы которого лежат в разных множествах. Пусть $x_{i_1}^{k_1} = 1, x_{i_2}^{k_2} = 0$. Тогда слагаемые для $j = j_1(e)$ и $j = j_2(e)$ равны $w^* + w_e/2$ и вес ребра входит в целевую функцию. Таким образом, значения целевых функций на решениях (x_i^k) и (V_1^x, V_2^x) совпадают с точностью до константы $W|V| + 2w^*|E|$.

Предположим, что разрез (V_1^x, V_2^x) не является локальным максимумом. Тогда найдется вершина $v \in V$, сдвиг которой из одного множества разбиения в другое приводит к росту целевой функции. В этом случае решение (x_i^k) не является локальным максимумом для окрестности N_1, так как сдвиг вершины v соответствует замене

$$x_{i_v}^k := 1 - x_{i_v}^k, \quad x_{i_{v'}}^k := 1 - x_{i_{v'}}^k \text{ при } k = k(i_v)$$

с тем же изменением целевой функции. Значит, (V_1^x, V_2^x) — локальный максимум, и построенное сведение является PLS сведением.

Убедимся, что это плотное сведение. В качестве множества R возьмем все допустимые решения (x_i^k), обладающие свойством: $x_{i_v}^{k(v)} + x_{i_{v'}}^{k(v)} = 1$ для любого $v \in V$. Ранее было показано, что это множество содержит все локальные максимумы задачи FLG, т. е. условие 1 определения 8 выполнено.

Проверим условие 2. Пусть (V_1, V_2) — произвольный разрез. Для k-го игрока найдем такую вершину v, что $i_v = k$, и положим

$$x_{i_v}^k = \begin{cases} 1, & \text{если вершина } v \text{ принадлежит } V_1, \\ 0 & \text{в противном случае,} \end{cases} \qquad x_{i'_v}^k = 1 - x_{i_v}^k, \qquad i_v = k.$$

При $i_v \neq k$ положим $x_{i_v}^k = x_{i'_v}^k = 0$. Получаем взаимно-однозначное соответствие между разрезами и элементами множества R.

Проверим последнее условие. Заметим, что в графе переходов задачи FLG нет дуг вида $(q_1, q_2), q_1 \in R, q_2 \notin R$, так как значение целевой функции для q_2 всегда меньше чем для q_1. Другими словами, любой путь, начинающийся и заканчивающийся в R, целиком лежит в R. Путь без промежуточных вершин из R может быть только дугой вида (q_1, q_2), $q_1, q_2 \in R$. Но такой дуге соответствует дуга в графе переходов задачи о максимальном разрезе. □

При доказательстве теоремы мы положили $f_i^k = 0$ для всех i, k. Следовательно, справедливо следующее утверждение.

Следствие 12 *Задача (FLG, N_1) является плотно PLS–полной для случая $f_i^k = 0, i \in I, k = 1, \ldots, p$.*

Рассмотрим теперь более общий случай, когда игроки могут открывать несколько предприятий. Цель состоит в исследовании вычислительной сложности задачи нахождения равновесий по Нэшу в этой более сложной игре. Покажем, что эта задача также является плотно PLS–полной. Для достижения этой цели снова перейдем к задаче поиска локальных максимумов относительно окрестности N_a и предъявим плотное PLS–сведение этой задачи к исходной задаче с окрестностью N_1.

Теорема 34 *Задача локального поиска (FLG, N_a), в которой k-й игрок может открывать не более $a_k \geq 1$ предприятий, является плотно PLS-полной.*

Доказательство. Покажем, что задача локального поиска (FLG, N_1) плотно PLS–сводится к задаче (FLG, N_a). Заметим сначала, что если взять в качестве функций h и g из определения PLS–сведения тождественные функции, то сразу получается PLS–полнота задачи (FLG, N_a). Однако это сведение не является плотным. Граф переходов новой задачи содержит больше дуг, чем исходный, и высота вершин может оказаться меньше высоты вершин в исходном графе.

Чтобы установить плотное PLS–сведение, добавим $(a_k - 1)$ дополнительных клиентов для каждого игрока и столько же дополнительных предприятий. Для каждого нового предприятия положим $f_i^k = 0$, если это предприятие введено для игрока k, и $f_i^k = M$, где M — достаточно большое число, если это не так. Прибыль $(\overline{c}_{ij})$ от обслуживания новых клиентов положим равной M для новых предприятий и равной 0 для

всех остальных. Легко заметить, что в такой задаче каждый игрок откроет в любом равновесии по Нэшу все $(a_k - 1)$ своих новых предприятий и будет выбирать последнее предприятие по тем же правилам, что и в задаче (FLG, N_1). □

5.4 Поиск приближенных равновесий

С практической точки зрения представляет интерес следующая модификация исходной игры. Мы неявно предполагаем, что игрок может менять свою стратегию бесплатно, т.е. перенос предприятия на новое место не требует предварительных расчетов, обоснований или других материальных затрат. Такое предположение упрощает модель, но делает ее беднее с содержательной точки зрения. Предположим, что такие затраты можно оценить, и для простоты будем задавать их величинами $\Delta_k, k = 1, \ldots, p$. Конечно, эта величина должна зависеть от текущей стратегии игрока и новой стратегии, которая приходит на смену старой. Однако пока мы не будем уточнять явный вид этой зависимости. Итак, если игрок k меняет свою стратегию, он несет расходы в размере Δ_k. Другими словами, игрок не будет менять свое решение, если приращение прибыли не превосходит данной величины. Если ни один из игроков не может увеличить свою прибыль при условии, что другие игроки не меняют свой выбор, то такое решение будем называть *приближенным равновесием* по Нэшу с погрешностью $\Delta = (\Delta_1, \ldots, \Delta_p)$. Исходные равновесия также будут и приближенными равновесиями при любом неотрицательном Δ. Задача поиска приближенных равновесий кажется более простой, поскольку число подходящих состояний увеличилось, а требуется найти хотя бы одно из них. Тем не менее, это не всегда так. Если Δ_k являются постоянными величинами, то поиск приближенных равновесий снова оказывается плотно PLS–полной задачей. Если же величины Δ_k могут быть оценены снизу функцией $\mu(i_1, \ldots, i_p)$, то есть $\Delta_k \geq \varepsilon\mu(i_1, \ldots, i_p)$ для любого $k = 1, \ldots, p$, то задача может быть решена с полиномиальной трудоемкостью.

Теорема 35 *Если $\Delta_k \geq \varepsilon\mu(i_1, \ldots, i_p)$ для любых стратегий $(i_1, \ldots, i_p)$, то приближенное равновесие с погрешностью $\Delta = (\Delta_1, \ldots, \Delta_p)$ может быть найдено за полиномиальное время от длины записи исходных данных и величины $1/\varepsilon$.*

Доказательство. Для поиска приближенных локальных оптимумов воспользуемся алгоритмом ε-Local Search из раздела 4.6. Применим его к задаче (FLG, N_1). Получим приближенный локальный максимум с относительной погрешностью ε, т.е. решение (x_i^k), удовлетворяющее условию $(1+\varepsilon)F(x_i^k) \geq F(\overline{x}_i^k)$ для всех $\overline{x}_i^k \in N_1(x_i^k)$. Это решение позволяет получить стратегии игроков $(i_1, \ldots, i_p)$: k-й игрок открывает предприятие i тогда и только тогда, когда $x_i^k = 1$. Полученные стратегии удовлетворяют условию

$$(1+\varepsilon)\mu(i_1, \ldots, i_p) \geq \mu(i_1, \ldots, i_{k-1}, i'_k, i_{k+1}, \ldots, i_p)$$

для любого $i'_k \neq i_k$ и $k = 1, \ldots, p$. Так как $\Delta \geq \varepsilon\mu(i_1, \ldots, i_p)$, то получается приближенное равновесие с погрешностью не более Δ. Трудоемкость процедуры получения таких равновесий определяется трудоемкостью алгоритма ε–Local Search. Число шагов этого алгоритма оценивается полиномом от длины записи исходных данных и величины $1/\varepsilon$. Так как каждый шаг алгоритма также является полиномиальным, то получаем требуемое. □

Вернемся теперь к ситуации, когда Δ_k являются постоянными величинами. В этом случае приближенным равновесиям будут соответствовать приближенные локальные оптимумы с абсолютным отклонением Δ_k, т.е. решения (x_i^k), удовлетворяющие условию:

$$F(x_i^k) + \Delta_k \geq F(\overline{x}_i^k)$$

для всех $\overline{x}_i^k \in N_1(x_i^k), k = 1, \ldots, p$. Интуитивно кажется, что нахождение таких приближенных локальных оптимумов проще задачи нахождения самих локальных оптимумов. Следующая теорема показывает, что это не так.

Теорема 36 *Если Δ_k являются постоянными величинами, то поиск приближенного равновесия является плотно PLS–полной задачей.*

Доказательство. Без ограничения общности можно считать, что все коэффициенты исходной задачи являются целыми числами и целевая функция принимает тоже только целочисленные значения. Рассмотрим граф переходов TG_Π задачи $\Pi = (FLG, N_1)$. Модифицируем его, удалив те дуги, для которых значения целевой функции на концах дуги отличаются не более чем на Δ_k. В модифицированном графе длина кратчайшего пути из заданной вершины в сток может не совпадать с длиной

кратчайшего пути в исходном графе TG_{Π}. Тем не менее, ниже будет показано, что для любой константы Δ_k и любой задачи (FLG, N_1) можно построить такой пример y' задачи нахождения приближенного локального максимума с константой Δ_k, что его модифицированный граф переходов будет совпадать с графом $TG_{\Pi}(y)$. Так как задача локального поиска $\Pi = (FLG, N_1)$ является плотно PLS–полной, то получаем требуемое.

Пусть y — пример задачи (FLG, N_1) с целочисленными значениями целевой функции $F(x, y)$ и коэффициентами $(\overline{c}_{ij}^k, f_i^k)$. Построим новый пример y' задачи FLG с тем же множеством допустимых решений и новой целевой функцией $F'(x_i^k, y')$, имеющей коэффициенты

$$\begin{cases} \overline{c}_{ij}^{k\,\prime} &= \overline{c}_{ij}^k(1+\Delta_k), \\ f_i^{k\,\prime} &= f_i^k(1+\Delta_k), \end{cases} \qquad i \in I, j \in J, k = 1, \ldots, p.$$

Заметим, что любой локальный максимум x_i^k для примера y является приближенным Δ–локальным максимумом для примера y', и наоборот. Действительно, если

$$F(x_i^k, y) - F(\overline{x}_i^k, y) < 1 \text{ для всех } \overline{x}_i^k \in N_1(x_i^k),$$

то

$$F'(x_i^k, y') - F'(\overline{x}_i^k, y') < (1+\Delta_k) \text{ для всех } \overline{x}_i^k \in N_1(x_i^k)$$

и x_i^k — приближенный локальный максимум. Граф переходов для примера y' и модифицированный граф переходов с константой $\Delta = (\Delta_1, \ldots, \Delta_p)$ совпадают друг с другом и с графом переходов для примера y. □

При доказательстве теоремы существенно использовался тот факт, что задача локального поиска (FLG, N_1) является плотно PLS–полной. Так как задача (FLG, N_a) тоже является таковой, то, повторяя рассуждения для этой более общей задачи, получаем следующее

Следствие 13 *Если Δ_k являются постоянными величинами, то задача поиска приближенных равновесных решений для случая, когда k-й игрок может открывать не более $a_k \geq 1$ предприятий, является плотно PLS–полной задачей.*

5.5 Сложность алгоритмов локального улучшения

Рассмотрим теперь вопрос о трудоемкости получения равновесных решений с помощью следующего итерационного алгоритма. Пусть некоторое решение (x_i^k) не является равновесным, т.е. существует игрок, быть может, не один, у которого можно увеличить его прибыль, при выборе других предприятий. Шаг алгоритма состоит в нахождении такого игрока и выборе для него новых предприятий с увеличением прибыли. Процедура выбора предприятий может быть любой. Наша цель состоит в оценке числа шагов такого итерационного процесса в худшем случае при любой процедуре выбора.

Теорема 37 *Итерационный алгоритм в худшем случае требует экспоненциального числа шагов для нахождения равновесного решения при любой процедуре выбора игрока и лучших предприятий для него.*

Справедливость утверждения следует из плотной PLS полноты задачи локального поиска (FLG, N_1) и существования задачи экспоненциальной высоты в классе PLS [212]. Оценка остаётся экспоненциальной при использовании любого правила выбора игрока и лучшего предприятия для него: детерминированного, вероятностного или любого другого произвольной сложности.

В задаче поиска равновесных решений нужно найти хотя бы одно (любое) равновесное решение. Однако, если решения игроков уже известны, и они готовы их менять только при увеличении прибыли, то возникает другая задача, более важная с практической точки зрения. Найти равновесное решение, достижимое итерационным алгоритмом локального улучшения из заданной начальной позиции. Другими словами, в графе переходов задана стартовая вершина. Требуется найти сток, достижимый из стартовой вершины вдоль некоторого ориентированного пути. Оказывается, такая задача является значительно более сложной, чем предшествующая. Из следующей теоремы получаем, что если для этой задачи удастся построить полиномиальный алгоритм, то полиномиально разрешимой будет любая задача из класса PSPACE и, в частности, все задачи из класса NP.

Теорема 38 *Нахождение равновесного решения при заданной начальной позиции игроков является PSPACE-полной задачей.*

Известно [212], что в классе PLS есть задачи локального поиска с фиксированной стартовой точкой, которые являются PSPACE-полными. Задача о максимальном разрезе графа с окрестностью *Flip* — одна из них. Установленная в теореме 33 плотная сводимость влечет полиномиальную сводимость соответствующих задач с фиксированными стартовыми решениями [212]. Следовательно, нахождение равновесного решения, достижимого из заданной начальной позиции игроков, является PSPACE-полной задачей. □

Глава 6

Электронная библиотека «Дискретные задачи размещения»

Создание электронных библиотек тестовых примеров является одним из бурно развивающихся направлений в области экспериментального исследования алгоритмов. Для задач дискретной оптимизации активно используются следующие библиотеки общего назначения:

– Operations Research Library (http://people.brunel.ac.uk/mastjjb/jeb/info.html),
– DIMACS Implemetation Challenge (http://dimacs.rutgers.edu/Challenges),

а также специализированные библиотеки:

– Traveling Salesman Problems Library (http://www.ing.unlp.edu.ar/cetad/mos/TSPBIB_home.html),
– Project Scheduling Problems Library (http://www.bwl.uni-kiel.de/Prod/psplib),
– Quadratic Assignment Problem Library (http://www.seas.upenn.edu/qaplib/),
– Library of Locations Algorithms (http://www.mathematik.uni-kl.de/ lola)

и др.

Библиотека Operations Research Library пользуется наибольшей популярностью и содержит наибольшее количество тестовых примеров по разным моделям исследования операций. В частности, в ней содержатся тесты и для дискретных задач размещения: задачи о покрытии, простейшей задачи размещения, задачи размещения с ограничениями на объемы производства продукции, обобщенной задачи о назначениях и др. К сожалению, многие из этих тестов, за исключением задачи о покрытии множествами, имеют небольшой разрыв двойственности. Исследование оптимизационных алгоритмов на этих тестах не всегда продуктивно. Они создают иллюзию, что даже при большой размерности задачи оптимальное решение может быть найдено сравнительно легко. Более того, указанные библиотеки содержат только оптимальные решения и исходные

данные. Они не дают никакой информации о разрыве целочисленности, числе оптимальных решений, сравнительных характеристик разных алгоритмов на указанных тестах и другой полезной информации, необходимой при экспериментальных исследованиях. В России работы над созданием таких библиотек только начинаются. Электронная библиотека «Дискретные задачи размещения» является одним из таких проектов, где сделана попытка подойти к данному вопросу на качественно новом уровне.

6.1 Структура библиотеки

Сложность решения многих задач дискретной оптимизации объясняется отчасти наличием большого числа локальных оптимумов, их непредсказуемым и, фактически, случайным расположением в допустимой области. Принадлежность некоторой задачи к классу NP-трудных задач, по сути, означает отсутствие "простых" правил для нахождения глобального оптимума. Строго доказать несуществование таких правил пока никому не удается. Вместе с тем, большинство точных методов при теоретически экспоненциальной трудоемкости на практике находят оптимальное решение сравнительно легко. Электронная библиотека «Дискретные задачи размещения» создана для проведения экспериментальных исследований. Она доступна в интернет по адресу http://math.nsc.ru/AP/benchmarks/ и представляет собой гипертекстовый документ, выполненный в формате HTML. Головная страница библиотеки содержит ссылки на основные разделы, посвященные следующим задачам:

– простейшая задача размещения;

– задача размещения с ограничениями на объемы производства;

– задача о p-медиане;

– задача о p-медиане с предпочтениями клиентов;

– конкурентная задача о p-медиане;

– задача о поставках.

Каждый раздел содержит:

– постановку задачи на содержательном уровне и точную математическую формулировку в терминах частично-целочисленного программирования;

– обзор литературы по точным и приближенным методам решения задачи. Наиболее популярным методам, таким как метод ветвей и границ, Лагранжевы релаксации, поиск с запретами, генетические алгоритмы и

алгоритмы имитации отжига уделено особое внимание, каждому из них посвящена отдельная страница с описанием и ссылками на литературу;

– тестовые примеры, трудные в вычислительном отношении, для нахождения как точного, так и приближенного решения с малой относительной погрешностью;

– результаты тестовых расчетов, содержащие точное решение и величину разрыва целочисленности.

Для демонстрации возможностей вышеупомянутых методов разработано программное обеспечение, которое на примере простейшей задачи размещения иллюстрирует их работу. Демонстрационная версия программного продукта снабжена комментариями и дает пользователю возможность влиять на процесс решения задачи с помощью управляющих параметров. Графические возможности системы позволяют увидеть изменение относительной погрешности в ходе работы алгоритмов и изменение расстояния Хэмминга от текущей точки до глобального оптимума, до нулевой точки и до точки, полученной рандомизированными жадными алгоритмами.

6.2 Простейшая задача размещения

Данный раздел библиотеки является самым большим. Здесь представлены восемь классов тестовых примеров, сильно отличающихся по трудоемкости нахождения точного решения:

– примеры, использующие конструкцию конечных проективных плоскостей (класс FPP);

– примеры, использующие совершенные коды с расстоянием 3 (класс BPC);

– примеры, базирующиеся на шахматных торах (класс CB);

– три класса примеров с большим разрывом целочисленности (классы GapA, GapB, GapC);

– примеры с матрицами расстояний на двумерной евклидовой плоскости (класс Euclidean);

– примеры с матрицей расстояний, порожденных случайным образом с равномерным распределением из заданного интервала (класс Uniform).

Первый класс является полиномиально разрешимым. Зная способ порождения примеров, легко найти точное решение задачи. Однако для методов локального поиска он оказывается достаточно трудным [186],

так как каждому пучку прямых соответствует локальный минимум задачи с большим бассейном притяжения.

Второй и третий классы интересны тем, что число строгих локальных оптимумов для них растет экспоненциально с ростом размерности задачи. Так как по построению любой из них может оказаться глобальным оптимумом, то процесс решения задачи может быть достаточно долгим для любого метода, в том числе и для метаэвристик.

Три класса с большим разрывом двойственности являются сложными для метода ветвей и границ [158, 3]. Разрыв составляет более 20%, что обуславливает порождение огромного дерева ветвлений с несколькими десятками миллионов вершин даже при небольшой размерности задачи, $n = m = 100$. Самый сложный из этих классов, *Gap–C*, является также трудным и для методов локального поиска: вероятностного поиска с запретами, генетического алгоритма и схемы GRASP. Частота нахождения такого решения не превышает 0,53 при выполнении этими алгоритмами 10^4 итераций, соответствующих переходу от текущего решения к соседнему для объединения окрестностей Flip и Swap. Последние два класса являются наиболее легкими. При заданной размерности $n = m = 100$ они легко решаются как методом ветвей и границ, так и метаэвристиками. Приведем точное описание этих классов.

6.2.1 Полиномиально разрешимые примеры

Рассмотрим конечную проективную плоскость порядка k [140]. Она состоит из $n = k^2 + k + 1$ точек $x_1, \ldots, x_n$ и такого же количества прямых $L_1, \ldots, L_n$. Каждая прямая есть набор точек, и $0-1$ матрица инциденций A задает вхождение точек в состав прямых: $a_{ij} = 1 \Leftrightarrow x_j \in L_i$. Матрица A обладает следующими свойствами:

1. каждая строка содержит ровно $k + 1$ единицу;
2. каждый столбец содержит ровно $k + 1$ единицу;
3. скалярное произведение любых двух строк равно единице;
4. скалярное произведение любых двух столбцов равно единице.

Такие матрицы существуют, если k — степень простого числа. Множество прямых, проходящих через заданную точку, называется пучком прямых. Мощность каждого пучка равна $k + 1$, и пересечение любых двух пучков состоит ровно из одной прямой.

Определим семейство примеров простейшей задачи размещения сле-

дующим образом. Положим $I = J = \{1, \ldots, n\}$ и

$$g_{ij} = \begin{cases} \xi_{ij}, & \text{если } a_{ij} = 1 \\ M, & \text{если } a_{ij} = 0 \end{cases}, \qquad f_i = f > \sum_{i \in I} \sum_{j \in J} \xi_{ij},$$

где ξ_{ij} — случайные числа, взятые из заданного интервала, M — достаточно большое число, $M > nf$. Это семейство обозначим через FPP_k. Так как $f > \sum_{i \in I} \sum_{j \in J} \xi_{ij}$, то оптимальному решению соответствует минимальное число прямых, покрывающих все точки. Легко проверить, что это пучок. Пучков ровно n штук, следовательно, оптимальное решение может быть найдено за полиномиальное время. Каждому пучку соответствует строгий локальный минимум в задаче UFLP с окрестностями Flip, Swap и их объединением. Случайные числа ξ_{ij} определяют глобальный оптимум на множестве пучков. Расстояние Хэмминга между этими строгими локальными минимумами ровно $2k$. Значит, диаметр области, где расположены локальные оптимумы, не меньше $2k$ и растет с ростом размерности задачи. Более того, численные эксперименты показывают, что такие строгие локальные минимумы имеют большую зону притяжения, что создает дополнительные трудности для методов локального поиска. Для поиска с запретами приходится использовать высокую рандомизацию окрестности или большие списки запретов. Для алгоритмов имитации отжига — высокую стартовую температуру. Если популяция в генетическом алгоритме состоит только из локальных минимумов, соответствующих пучкам прямых, то оператор скрещивания порождает решения с большими значениями целевой функции и процедуры локального спуска с такими стартовыми решениями оказываются малоэффективными. Для вероятностных жадных стратегий (GRASP) это семейство тоже является сложным [185, 186].

6.2.2 Примеры с экспоненциальным числом строгих локальных минимумов

Рассмотрим два класса примеров, в которых число строгих локальных минимумов растет экспоненциально с ростом размерности задачи. Первый класс BPC использует конструкцию совершенных бинарных кодов с расстоянием 3. Второй класс CB — шахматные доски, свернутые в тор.

Обозначим через B^k множество всех $0-1$ векторов длины k. Совершенным бинарным кодом с расстоянием 3 называют такое подмножество

$C \subset B^k$ мощности $2^k/(k+1)$, что расстояние Хэмминга $d(c_1c_2)$ для любых кодовых слов $c_1, c_2 \in C, c_1 \neq c_2$ всегда не меньше трех. Такие коды существуют для $k = 2^r - 1$, где r целое число, большее единицы. Построим пример исходных данных простейшей задачи размещения по следующему правилу. Положим $n = 2^k, I = J = \{1, \ldots, n\}$. Каждому элементу $i \in I$ соответствует вершина $x(i)$ гиперкуба. Пусть $(\xi_{ij}), i \in I, j \in J$, — некоторая матрица, порожденная, например, случайным образом с элементами из заданного интервала. Положим

$$c_{ij} = \begin{cases} \xi_{ij}, & \text{если } d(x(i), x(j)) \leq 1, \quad i \in I, j \in J, \\ +\infty & \text{в противном случае.} \end{cases}$$

Известно, что любой совершенный бинарный код C с расстоянием 3 порождает разбиение гиперкуба на $2^k/(k+1)$ непересекающихся шаров радиуса 1. Такое разбиение соответствует строгому локальному минимуму в задаче размещения с окрестностью *Flip* $\cup$ *Swap*. Число совершенных кодов $N(k)$ растет экспоненциально с ростом k и удовлетворяет соотношению [45]

$$\mathcal{N}(k) \geq 2^{2^{\frac{k+1}{2}\log_2(k+1)}} \cdot 3^{2^{\frac{k-3}{4}}} \cdot 2^{2^{\frac{k+5}{4}\log_2(k+1)}}.$$

Минимальное расстояние между двумя совершенными кодами (строгими локальными оптимумами) не меньше $2^{(k+1)/2}$, то есть также растет с ростом размерности задачи.

Рассмотрим теперь шахматную доску размером $3k \times 3k, k \geq 3$, и склеим её границы так, чтобы получился тор. Каждая клетка тора имеет 8 соседних клеток. Например, клетка (1,1) имеет соседей: $(1,2)$, $(1,3k)$, $(2,1)$, $(2,2)$, $(2,3k)$, $(3k,1)$, $(3k,2)$, $(3k,3k)$. Положим $n = 9k^2, I = J = \{1, \ldots, n\}$,

$$c_{ij} = \begin{cases} \xi_{ij}, & \text{если клетки } i, j \text{ — соседние,} \\ +\infty & \text{в противном случае,} \end{cases} \qquad i \in I, j \in J,$$

$$f_i = f \geq \sum_{i \in I} \sum_{j \in J} \xi_{ij}, \quad i \in I.$$

Каждая клетка вместе с ее соседями образует квадрат 3×3. Тор разбивается на k^2 таких непересекающихся квадратов. Каждое такое разбиение тора соответствует строгому локальному минимуму в задаче размещения с окрестностью Flip $\cup$ Swap. Таких строгих локальных минимумов

$2 \cdot 3^{k+1} - 9$. Минимальное расстояние между ними равно $2k$. Таким образом, число строгих локальных минимумов растет экспоненциально с ростом размерности задачи. Минимальное расстояние между ними также возрастает.

6.2.3 Примеры с большим разрывом целочисленности

Как мы увидим позже, разрыв целочисленности для приведенных выше классов BPC, CB, FPP достаточно мал. Метод ветвей и границ [158] находит оптимальное решение достаточно легко. Поэтому представляет интерес построение примеров, которые были бы трудны не только для локальных алгоритмов, но и для точных методов.

Предшествующие классы обладают следующим свойством: в матрице (c_{ij}) каждый столбец и каждая строка имеют одинаковое число небесконечных элементов. Обозначим это число через l. Величину l/n назовем плотностью матрицы. Приведем алгоритм нахождения случайных матриц (c_{ij}) с заданной плотностью.

Генератор случайных матриц (l, n)

1. $J \leftarrow \{1, \ldots, n\}$;
2. Column $[j] \leftarrow 0$ for all $j \in J$;
3. $c[ij] \leftarrow +\infty$ for all $i, j \in J$;
4. **for** $i \leftarrow 1$ **to** n
5. **do** $l_0 \leftarrow 0$
6. **for** $j \leftarrow 1$ **to** n
7. **do if** $n - i + 1 = l -$Column$[j]$
8. **then** $c[ij] \leftarrow \xi[ij]$
9. $l_0 \leftarrow l_0 + 1$
10. Column$[j] \leftarrow$ Column$[j] + 1$
11. $J \leftarrow J \setminus j$
12. выбрать подмножество $j' \subset J, |J'| = l - l_0$ случайным образом и положить

$c[ij] \leftarrow \xi_{ij}$; Column$[j] \leftarrow$ Column$[j] + 1$, для всех $j \in J'$.

Вспомогательный вектор Column$[j]$ содержит текущее значение числа малых элементов в j-м столбце генерируемой матрицы. Идентификатор l_0 используется для подсчета столбцов матрицы, где малые элементы

непременно должны быть размещены в i-й строке. Такие столбцы выявляются заранее (строка 7 алгоритма) и удаляются из множества J (строка 11). Заметим, что удаление строк 6–11 из алгоритма позволяет генерировать случайные матрицы, у которых ровно l малых элементов в каждой строке. Транспонированием матрицы можно получить аналогичное свойство для столбцов. Таким образом, получаем три класса тестовых примеров:

Gap–A: каждый столбец матрицы (c_{ij}) имеет ровно l малых элементов.

Gap–B: каждая строка матрицы (c_{ij}) имеет ровно l малых элементов.

Gap–C: каждый столбец и каждая строка матрицы (c_{ij}) имеют ровно l малых элементов.

Для этих классов по-прежнему $I = J = \{1, \ldots, n\}$ и $f_i = f \geq \sum_{i \in I} \sum_{j \in J} \xi_{ij}$.

Обозначим через δ разрыв целочисленности:

$$\delta = \frac{F_{opt} - F_{LP}}{F_{opt}} \cdot 100\%,$$

где F_{opt} — оптимальное значение задачи, F_{LP} — значение линейной релаксации. Для $l = 10, n = 100$ на классе Gap–C величина δ оказывается в интервале от 21% до 29%. Как следствие, метод ветвей и границ требует громадного числа итераций, до $0,5 \cdot 10^9$, и далеко не всегда находит оптимум за первую половину потраченного времени.

6.2.4 Поведение метаэвристик

Для исследования поведения метаэвристик на введенных классах данных были сформированы тестовые примеры, по 30 в каждом классе. Значения ξ_{ij} выбирались из множества $\{0, 1, 2, 3, 4\}$ случайным образом с равномерным распределением. Значение f полагалось равным 3000. Все примеры доступны в интернет. Оптимальные значения найдены методом ветвей и границ. Таблица 10 представляет среднее значение характеристик этого метода на указанных классах. Класс BPC_7 соответствует значению $k = 7$ и $n = 2^k = 128$. Класс CB_4 строился для $k = 4, n = 9k^2 = 144$. Класс FPP_{11} соответствует $k = 11, n = k^2 + k + 1 = 133$. Для сравнения в таблице приведены два других уже упоминавшихся класса Unifirm, где $c_{ij} \in [0, 10^4]$, и Euclidean, где матрица c_{ij} соответствует

евклидовым расстояниям между точками двухмерной плоскости. Точки выбирались случайным образом с равномерным распределением из квадрата 7000×7000 независимо друг от друга.

Таблица 10. *Храктеристики метода ветвей и границ*

Классы примеров	n	δ	Итерации	Рекордная итерация	Время счета
BPC_7	128	0,1	374 264	371 646	00:00:15
CB_4	144	0,1	138 674	136 236	00:00:06
FPP_{11}	133	7,5	6 656 713	6 635 295	00:05:20
GAP–A	100	25,6	10 105 775	3 280 342	00:04:52
GAP–B	100	21,2	30 202 621	14 656 960	00:12:24
GAP–C	100	28,4	541 320 830	323 594 521	01:42:51
Uniform	100	4,7	9 834	2 748	$<$ 00:00:01
Euclidean	100	0,1	1 084	552	$<$ 00:00:01

Столбец «Время счета», указывает среднее время решения задачи на PC Pentium 1200 MHz, RAM 128 Mb. Столбец «Итерации» показывает среднее число итераций метода. Столбец «Рекордная итерация» дает среднее значение числа итераций, необходимых для нахождения оптимума. Как видно из таблицы, классы GAP–A, GAP–B, GAP–C имеют наибольший разрыв целочисленности и являются наиболее трудными для метода ветвей и границ. Классы BPC_7, CB_4 имеют малый разрыв целочисленности. Однако они оказались значительно труднее классов Uniform и Euclidean. Число итераций для классов BPC_7, CB_4 на два порядка больше, чем для классов *Uniform* и *Euclidean*. Это может показаться странным, но такой эффект легко объясняется строением самих классов BPC и CB. Они действительно имеют большое число локальных оптимумов, мало отличающихся по целевой функции от глобального оптимума. Чтобы их отсечь, требуется значительное число итераций. Как мы увидим ниже, классы *Uniform* и *Euclidean* сильно отличаются от остальных классов по числу локальных оптимумов и их распределению в допустимой области. Они гораздо беднее и потому проще для решения.

Для того, чтобы понять различие между классами с точки зрения распределения локальных оптимумов в допустимой области, был проведен следующий эксперимент. Для примеров из каждого класса генерировалось 9000 допустимых решений задачи. Они порождались независимо

друг от друга с помощью датчика псевдослучайных чисел с равномерным распределением. К каждому решению применялся алгоритм локального улучшения с окрестностью $Flip \cup Swap$. В итоге получался набор локальных оптимумов. Часть из них совпадало между собой. Разительное отличие между классами связано, в первую очередь, с мощностью множества получаемых локальных оптимумов. Классы *Uniform* и *Euclidean* имеют патологически малые множества локальных оптимумов и, как будет показано ниже, являются простыми для метаэвристик. В Таблице 11 приводятся характеристики полученных множеств локальных оптимумов. Столбец «N» показывает мощность множеств для различных классов. Столбец «Диаметр» дает нижнюю оценку диаметра области, в которой располагаются локальные оптимумы. Эта величина вычислялась как максимальное попарное расстояние между полученными локальными минимумами.

Таблица 11. *Храктеристики множеств локальных оптимумов*

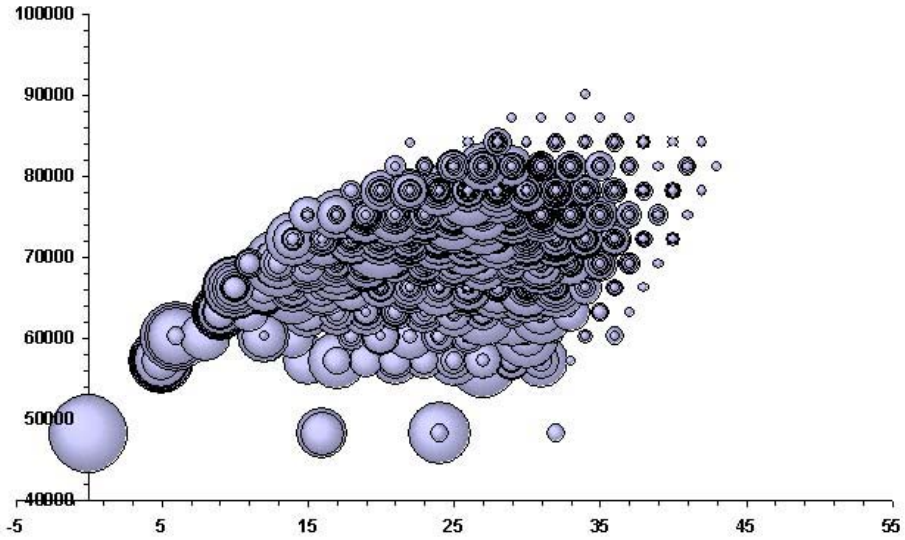
Классы примеров	N	Диаметр	Радиус			R_{100}	R^*
			min	ave	max		
BPC_7	8868	55	1	3	357	24	52
CB_4	8009	50	1	13	178	78	53
FPP_{11}	8987	51	1	2	8	3	1
GAP–A	6022	36	1	53	291	199	7
GAP–B	8131	42	1	180	164	98	16
GAP–C	8465	41	1	14	229	134	21
Uniform	1018	33	1	31	101	61	1
Euclidean	40	21	11	13	18	–	10

Рисунки 20a – 20$з$ показывают значения целевой функции задачи для разных локальных минимумов и их расстояние до глобального минимума. Для каждого локального минимума нарисован шар. Центр шара имеет координаты (x, y), где x — расстояние Хэмминга до глобального минимума, y — значение целевой функции. Радиусом шара служит число найденных локальных минимумов, обнаруженных рядом с данным, точнее, на расстоянии не более 10. В таблице 11 столбцы «min», «ave», «max» показывают минимальные, средние и максимальные радиусы полученных шаров в каждом классе. Класс FPP_{11} имеет очень малый средний и максимальный радиусы. Следовательно, локальные оптимумы имеют

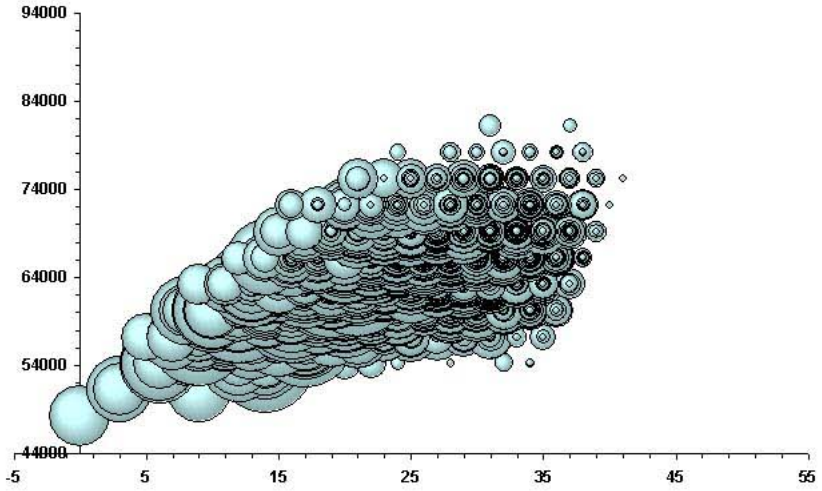
достаточно большой бассейн притяжения.

На рис. 20*в* локальные минимумы, соответствующие пучкам прямых, показаны двумя самыми нижними шарами. Один из них соответствует оптимальному решению задачи, $x = 0$. Второй — всем остальным пучкам. Расстояние между каждой парой пучков равно 22. Интересно отметить, что ближайший к глобальному оптимуму соседний локальный оптимум находится на расстоянии 12, достаточно далеко. По-видимому, каждый пучок прямых порождает достаточно "глубокую воронку" , попадая в которую, трудно выбраться, используя локальные методы. Столбец «R^*» в таблице 11 показывает радиус шара для глобального оптимума. Он равен 1 для классов $\mathrm{FPP}_1 1$ и *Uniform*. Максимальные значения 53 и 52 соответствуют классам CB_4 и BPC_7. Заметим, что для всех классов радиус глобального оптимума не является максимальным или минимальным. Скорее всего нет никакой корреляции между плотностью расположения локальных оптимумов в допустимой области и значением целевой функции.

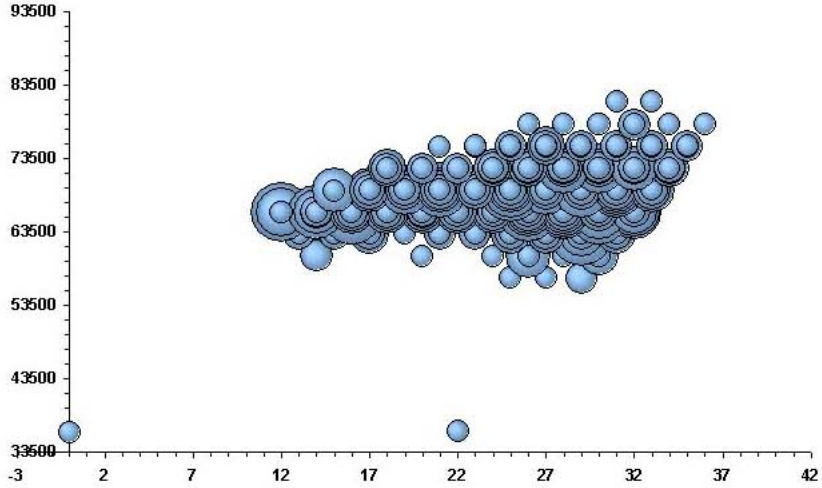
Столбец «R_{100}» показывает минимальный радиус для 100 самых больших шаров. Этот столбец указывает, что классы GAP–A, GAP–B, GAP–C имеют много шаров большого радиуса. Многие локальные минимумы являются нестрогими и образуют своего рода "плато" . Попадая на них, методы локального поиска часто теряют эффективность. Класс FPP_{11} имеет значение R_{100} равное 3, то есть все шары имеют малый радиус. Класс FPP_{11} является трудным, но по другим причинам. Там нет "плато".



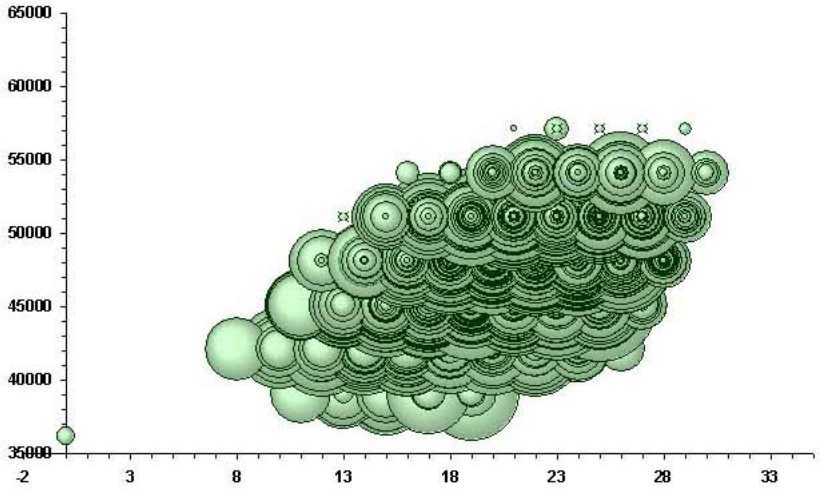
а) класс BPC_7



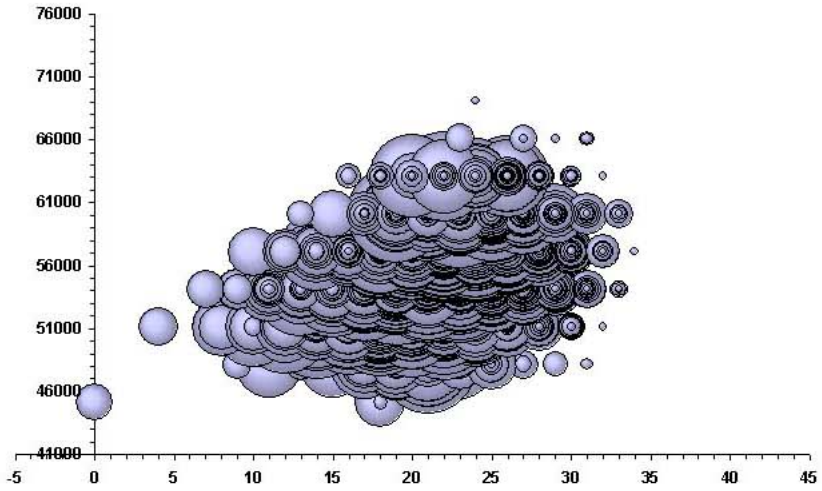
б) класс CB_4



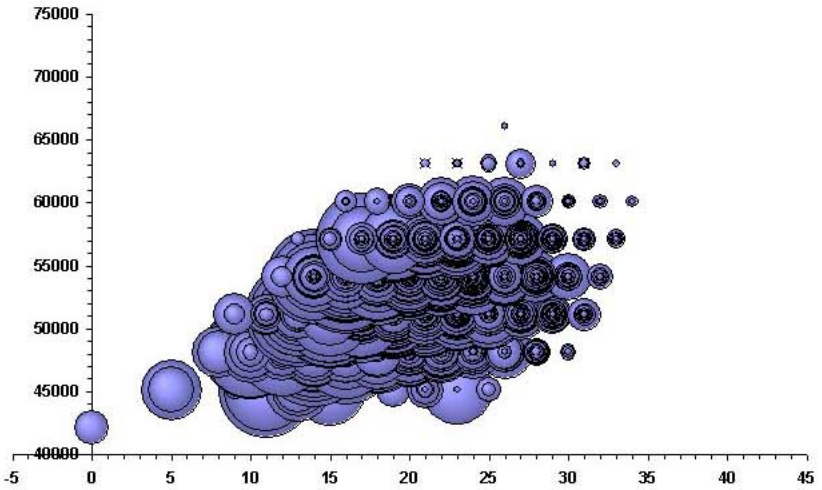
в) класс FPP_{11}



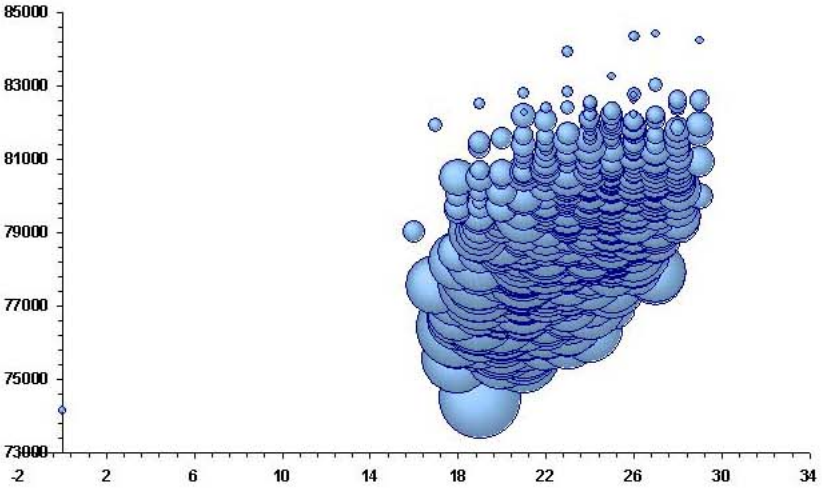
г) класс GAP–A



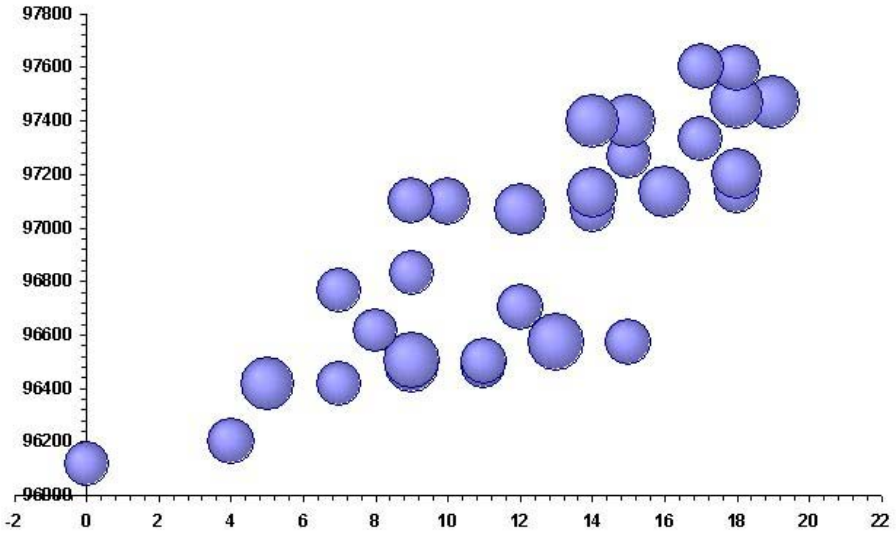
д) класс GAP–B



е) класс GAP–C



ж) класс Uniform



з) класс Euclidean

Рис 20. *Расположение локальных минимумов*

На рис. 20*а*, 20*б*, 20*е*, 20*ж* можно увидеть много малых шаров на большом удалении от глобального оптимума. Если представить множество локальных оптимумов в виде галактики, то на её границах будут шары малого радиуса, а в центральной части — много больших шаров. Область высокой концентрации локальных оптимумов содержит шары с высоким и низким значениями целевой функции. Глобальный оптимум может располагаться как в центральной части галактики (рис. 20*а*, 20*б*), так и далеко от центра и иметь малый радиус (рис. 20*в*, 20*ж*). По-видимому, нельзя предсказать его радиус и место расположения. Можно только отметить, что легкие классы имеют мало локальных оптимумов, например, $N = 1018$ для класса *Uniform* и $N = 40$ для класса *Euclidean*. Трудные классы имеют много локальных оптимумов, $N \approx 9000$ для классов BPC_7, FPP_{11}, GAP–C. Возможно, наиболее трудные примеры имеют несколько галактик локальных оптимумов, расположенных на достаточно большом расстоянии друг от друга и отделённых областью с высокими значениями целевой функции. Такие примеры представляют несомненный интерес для экспериментальных исследований. Один из таких классов для многостадийной задачи размещения представлен в следующем разделе этой главы.

Таблица 12 показывает частоту нахождения оптимального решения метаэвристиками: вероятностный поиск с запретами (PTS), генетический локальный поиск (GA), жадные эвристики с локальной оптимизацией (GRASP + LI). Все алгоритмы используют окрестность FLIP $\cup$ Swap.

Критерием остановки служит выполнение 10^4 шагов, соответствует переходу от текущего решения к соседнему. Таблица 6.3 подтверждает, что классы *Uniform* и *Euclidean* являются простыми для метаэвристик. Для нахождения оптимума любой из них потребовал не более 10^3 шагов.

Таблица 12. *Частота нахождения оптимальноrо решения*

Классы примеров	Диаметр	PTS	GA	GRASP + LI
BPC_7	128	0,93	0,90	0,99
CB_4	144	0,00	0,88	0,68
FPP_{11}	133	0,67	0,46	0,99
GAP–A	100	0,85	0,76	0,87
GAP–B	100	0,59	0,44	0,49
GAP–C	100	0,53	0,32	0,42
Uniform	100	1,0	1,0	1,0
Euclidean	100	1,0	1,0	1,0

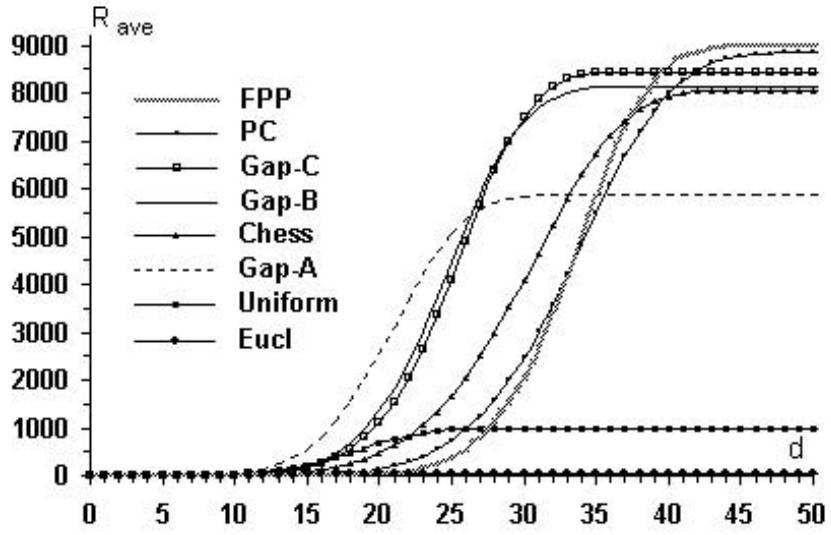


Рис. 21. *Изменение среднего радиуса* R_{ave}

Рис. 21 показывает изменение среднего радиуса R_{ave} с ростом расстояния d. Для каждого класса эта функция имеет три ярко выраженных участка: пологий, крутого подъема и снова пологий. Первый из них соответствует ситуации, когда шары мало пересекаются друг с другом. Последний участок соответствует значениям d, близким к диаметру той

части допустимой области, в которой находятся все найденные локальные минимумы. Точки перехода от второго участка к третьему показывают размеры области, содержащей основную часть локальных оптимумов. Для класса GAP–C эта точка примерно равна 33, в то время как для классов FPP_{11}, BPC_7, CB_4 она примерно равна 42. Следовательно, эта область для класса GAP–C существенно меньше, чем для классов FPP_{11}, BPC_7, CB_4. Столбец "Диаметр" в таблице 11 приводит к тем же выводам. Тем не менее, класс GAP–C более труден. Он имеет ту же плотность, но матрица (c_{ij}) порождалась случайным образом. По-видимому, это свойство является важным для генерации трудных примеров для простейшей задачи размещения.

6.3 Примеры с кластеризацией локальных минимумов

В настоящем разделе приводится семейство исходных данных многостадийной задачи размещения, для которых локальные оптимумы группируются в три кластера. Один кластер характеризуется большими значениями целевой функции, два других — малыми значениями. В задаче имеется один глобальный минимум. Чтобы попасть к нему локальными изменениями решений из другого кластера, требуется сначала посетить кластер с большими значениями целевой функции. Построенное семейство исходных данных может оказаться своего рода "ловушкой" для таких метаэвристик, как вероятностный поиск с запретами, генетические алгоритмы, локальный поиск с чередующимися окрестностями и др., если в алгоритмах не предусмотрены специальные процедуры диверсификации поиска.

Напомним постановку многостадийной задачи размещения. Пусть $I = \{1, \ldots, \mathbf{I}\}$ множество возможных пунктов размещения предприятий и $J = \{1, \ldots, \mathbf{J}\}$ множество потребителей. В каждом пункте $i \in I$ можно разместить только одно предприятие. Далее, где это не вызывает путаницы, будем отождествлять предприятие и пункт его размещения. Будем предполагать, что предприятия имеют узкую специализацию и для производства продукции требуются наборы предприятий разной специализации. Продукция, прежде чем попасть к потребителю, должна пройти несколько стадий обработки на специализированных предприя-

тиях [203]. Будем считать, что известны технологические цепочки вида $p = \{i_1^p, \ldots, i_{s(p)}^p\}$, возможно разной длины, согласно которым продукция последовательно проходит обработку на предприятиях i_1^p, $i_2^p, \ldots, i_{s(p)}^p$. В одной цепочке предприятия могут повторяться. Множество различных цепочек обозначим через $P = \{1, \ldots, \mathbf{P}\}$ и положим $P_i = \{p \in P | i \in p\}, i \in I$. Стоимость открытия i-го предприятия обозначим через $f_i \geq 0$, а стоимость производства продукции по технологической цепочке p и её транспортировки к j-му потребителю обозначим через $c_{pj} \geq 0$.

Введем переменные задачи:

$$x_p = \begin{cases} 1, & \text{если продукция выпускается по технологической цепочке } p, \\ 0 & \text{в противном случае.} \end{cases}$$

Многостадийная задача размещения состоит в отыскании такого варианта размещения предприятий и выбора на их основе подмножества технологических цепочек, чтобы с минимальными суммарными затратами удовлетворить запросы всех потребителей. С использованием введенных обозначений данная задача может быть записана следующим образом [21]: найти минимум функции

$$F(x) = \sum_{i \in I} f_i \max_{p \in P_i} \{x_p\} + \sum_{j \in J} \min_{p \in P} \{c_{pj} | x_p = 1\}.$$

Первое слагаемое задает стоимость открытия предприятий. Второе слагаемое задает стоимость производства и транспортировки продукции.

Рассмотрим семейство исходных данных $\mathcal{F}(n, m, K)$ многостадийной задачи размещения с $\mathbf{P} = \mathbf{J} = n, \mathbf{I} = m, m \leq n$ и мощностью допустимых решений не менее K, т. е. $\sum_{p \in P} x_p \geq K$. Построим по этому семейству новое семейство $\overline{\mathcal{F}}(2n, m+2, K)$ по следующему правилу. Положим $\overline{I} = I \bigcup \{m+1, m+2\}$ и

$$\overline{f}_i = \begin{cases} f_i, & \text{если } i \in I, \\ M_1 & \text{если } i = m+1, \\ M_2 & \text{если } i = m+2, \end{cases}$$

где $M_2 > M_1 \gg f_i > 0$, $i \in I$. Элемент $i_1 = m+1$ будет входить в оптимальное решение задачи и определять первый кластер локальных оптимумов. Элемент $i_2 = m+2$ будет определять второй кластер, где нет оптимальных решений. Локальные оптимумы, включающие оба элемента, будут формировать третий кластер с большими значениями целевой функции.

Множество $\overline{P}$ будет состоять из двух копий P_1 и P_2 множества P. В состав каждой цепочки из P_1 включим элемент i_1. В состав каждой цепочки из множества P_2 включим элемент i_2. Матрицу $\overline{c}_{pj}$ определим как матрицу, составленную подходящим образом из двух копий матрицы (c_{pj}) при $\overline{J} = J$.

Пусть (x_p) — допустимое решение для некоторого примера из семейства $\mathcal{F}(n, m, K)$. Определим по нему начальное решение для соответствующего примера из $\overline{\mathcal{F}}(2n, m+2, \widetilde{K})$ следующим образом:

$$\overline{x}_p = \begin{cases} 0, & \text{если } p \in P_1, \\ x_p, & \text{если } p \in P_2. \end{cases}$$

Для того чтобы получить оптимальное решение задачи, применим вероятностный алгоритм поиска с запретами [21]. Напомним, что в этом итерационном алгоритме число шагов может быть сколь угодно большим. На каждом шаге осуществляется переход от текущего решения к соседнему относительно заранее заданной окрестности. Обозначим через $d(x, y)$ расстояние Хэмминга между x и y. Через $N_k(x)$ обозначим окрестность точки x радиуса k в гиперкубе E^n, т. е.

$$N_k(x) = \{y \in E^n \mid d(x, y) \leq k\},\ k \geq 1.$$

Далее будем рассматривать только случай $k = 2$, хотя все рассуждения можно обобщить и для произвольного k. Для алгоритма поиска с запретами переход от точки x^{t-1} к x^t связан с изменением некоторых координат (i_t, j_t), а список запретов φ^t содержит такие пары за последние l шагов алгоритма:

$$\varphi^t = \{(i_t, j_t), (i_{t-1}, j_{t-1}), \ldots, (i_{t-l+1}, j_{t-l+1})\}.$$

Для удобства будем считать, что $i_t \leq j_t$. Если на t-м шаге менялось значение только одной координаты, то $i_t = j_t$. По построению в списке запретов все пары разные, и пара $(i, j), i \neq j$, не запрещает движение по парам $(i, i), (j, j)$ и наоборот. При $k > 2$ аналогичным образом строятся тройки координат, четверки и т.д. Список запретов удаляет из окрестности $N_2(x^t)$ ровно l точек. Множество незапрещенных точек обозначим через $N_k(x^t, \varphi^t)$. Для того, чтобы поиск был эффективным, целесообразно использовать малые значения l и управлять данным параметром в ходе работы алгоритма.

Для детерминированной окрестности алгоритм поиска с запретами при $l < n(n-1)/4$ будет порождать точки вида

$$x_p^t = \begin{cases} 0, & \text{если } p \in P_1, \\ 0 \text{ или } 1, & \text{если } p \in P_2. \end{cases} \qquad (*)$$

Другими словами, последовательность $\{x^t\}$ не уйдет за пределы множества P_2 и оптимальное решение никогда не будет получено. При $q < 1$ и $l < n(n-1)/4$ вероятность найти точное решение задачи стремится к 1 при неограниченном росте числа итераций, что связано с существованием пути из любого допустимого решения в глобальный оптимум в соответствующем графе соседства. Однако для данного семейства любой такой путь, начинающийся с решений вида (*), должен проходить через решения из третьего кластера с большими значениями целевой функции. Для того чтобы сделать первый шаг в таком направлении, рандомизированная окрестность не должна содержать точек вида (*), либо такие точки должны быть запрещены списком запретов. Вероятность такого события достаточно мала. Следовательно, вероятность выхода из второго кластера еще меньше и вероятность нахождения оптимального решения может оказаться ничтожно малой величиной. Аналогичной конструкцией можно добиться скопления локальных оптимумов и в бóльшее число кластеров.

Для исследования реального поведения алгоритмов были сформированы примеры следующей размерности:

$$|J| = 100,\ |I| = 50,\ |P| = 100.$$

Каждая цепочка из множества P содержит ровно 5 предприятий, $s(p) = 5$, и $|P_1| = |P_2|$, $f_i = 3000, i \in I$, $M_1 = M_2 = 20000$. Элементы матрицы (c_{pj}) задаются как евклидовы расстояния между случайно выбранными точками $j_1, j_2, \ldots, j_n$ из квадрата со стороной 1000. Сформированный класс является легким для метода ветвей и границ [19]. В среднем алгоритму требуется 780 итераций. Время счета на PC Intel Celeron 800 Mhz, RAM 128 Mb составляет 45,3 секунды. Из 780 итераций в среднем для нахождения оптимального решения требуется 116 итераций. Остальное время используется для доказательства оптимальности полученного решения.

Вероятностный алгоритм поиска с запретами также легко находит оптимальное решение задачи, если стартовая точка принадлежит кластеру

с глобальным оптимумом. Если же стартовая точка удовлетворяет условию (*), то последовательность $\{x^t\}$ не уходит из своего кластера за 10^6 итераций даже при высокой рандомизации окрестности, $q < 0,1$.

Для визуализации взаимного расположения кластеров был проведен следующий эксперимент на одном из тестовых примеров. Случайным образом с равномерным распределением на гиперкубе выбрано 9000 стартовых точек. Для каждой из них применялся детерминированный алгоритм локального спуска по окрестности N_2. Для полученных локальных оптимумов подсчитано расстояние Хэмминга до оптимального решения задачи x^*. На рис. 22 показаны результаты этого эксперимента.

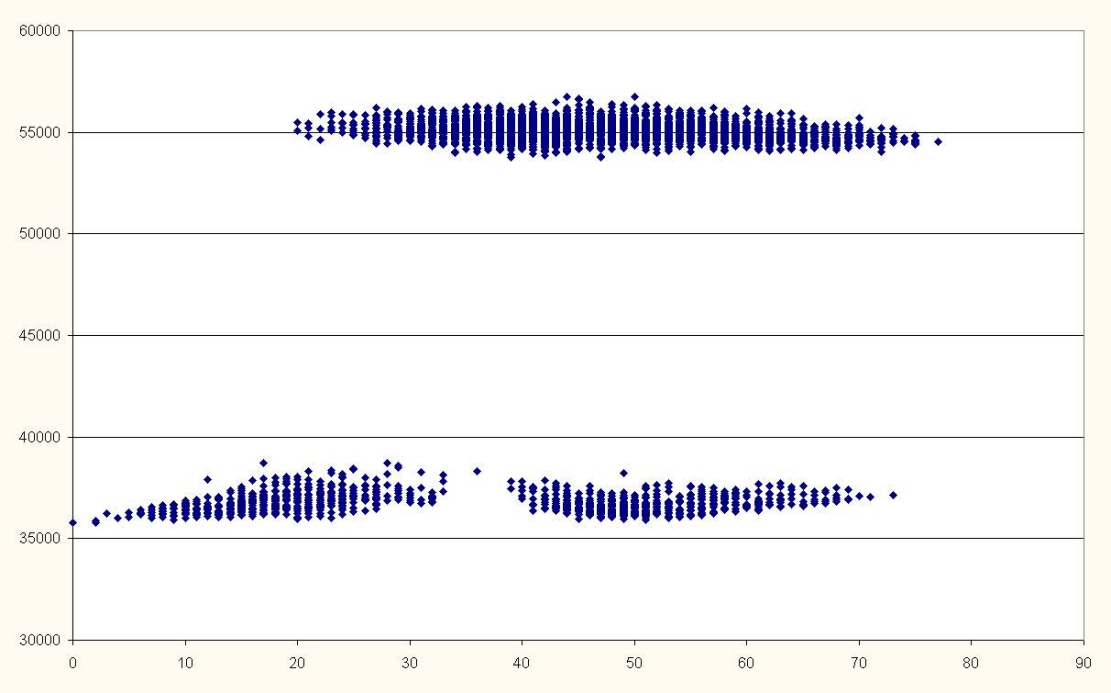


Рис. 22. *Взаимное расположение кластеров*

Локальные оптимумы группируются в три кластера. Один из них имеет большие значения целевой функции, порядка 55000. Два других имеют значительно меньшие значения, около 38000. Эти кластеры четко отделены друг от друга. Аналогичный эффект с кластеризацией локальных оптимумов может наблюдаться и в других задачах, например, при размещении буферных накопителей в гибких производственных системах [57, 104].

6.4 Примеры для конкурентной задачи о p-медиане

В разделе 1.3 рассматривалась задача об антагонистических размещениях — конкурентная задача о p-медиане или, следуя Хакими [139], задача о (r, p)-центроиде. Как уже отмечалось во введении, она принадлежит классу NP^{NP} (в полиномиальной иерархии — класс $\sum_2^P$) и является значительно более сложной в вычислительном отношении, чем любая задача из класса NP. В разделе 3.5.2 для ее точного решения предложен метод генерации переменных и ограничений. Он является конечным и гарантирует получение не только оптимального значения целевой функции, но и самого решения, на котором этот оптимум достигается. Метод предлагает точное решение задачи Конкурента (шаг 3) при заданном ходе Лидера и нахождение максимального дохода Лидера при заданном семействе решений Конкурента (шаг 2).

Для тестирования возможностей такого подхода сформированы примеры конкурентной задачи о p-медиане следующей размерности: $n = m \leq 100$. Матрица расстояний (c_{ij}) порождалась евклидовыми расстояниями между случайно выбранными точками на двухмерной евклидовой плоскости. Как следует из раздела 6.2.4, это простой класс, что облегчает поиск решения Конкурента на шаге 2.

Исследование поведения точного метода показало, что в предложенном варианте он позволяет решать только задачи малой размерности, $n = m \leq 30$, $p = r \leq 10$. Финальное семейство $\mathcal{F}$ решений Конкурента получается небольшим, до 100 элементов, и расчеты на персональном компьютере PC Pentium Intel Core 2 с частотой 1,87 ГГц укладываются в несколько часов. При бóльших размерностях получаются значительно бóльшие семейства. Шаг 2 становится все более и более трудоемким, что и является основным препятствием для решения задач средней размерности.

Для повышения эффективности метода была предложена следующая модификация, направленная на сокращение трудоемкости шага 2. Напомним, что на этом шаге решается следующая задача (см. раздел 5.2.2):

$$\max W$$

при условиях

$$\sum_{j \in J} w_j u_{jy} \geq W, \quad y \in \mathcal{F};$$

$$u_{jy} \le \sum_{i \in I_j(y)} x_i, \quad j \in J, y \in \mathcal{F};$$

$$\sum_{i \in I} x_i = p;$$

$$u_{jy}, x_i \in \{0, 1\}, \quad i \in I, j \in J, y \in \mathcal{F}.$$

Переменные задачи по-прежнему имеют следующий смысл:

$$u_{jy} = \begin{cases} 1, & \text{если клиент } j \text{ обслуживается Лидером,} \\ & \text{а Конкурент использует решение } y \\ 0, & \text{если клиент } j \text{ обслуживается Конкурентом} \\ & \text{и Конкурент использует решение } y, \end{cases}$$

$$x_i = \begin{cases} 1, & \text{если Лидер открывает предприятие } i, \\ 0 & \text{в противном случае.} \end{cases}$$

Она является NP-трудной в сильном смысле и решается методом ветвей и границ. Известно, что этот метод сравнительно быстро находит оптимальное решение, но затем долго доказывает, что это действительно оптимум. На это и тратится основное время.

Предположим, что нам известно оптимальное решение W^* исходной задачи двухуровневого программирования. Тогда точный метод можно модифицировать следующим образом. Добавим в задачу поиска $W(\mathcal{F})$ дополнительное ограничение: $W \ge W^*$ и будем искать не оптимальное, а допустимое решение задачи. Такая замена приводит к резкому падению трудоемкости одной итерации. Не требуется доказывать оптимальность. Достаточно проверить допустимость. По прежнему трудной остается только последняя итерация, на которой система оказывается несовместной. Именно это обстоятельство, несовместность системы, является критерием остановки модифицированного метода. Его общая схема может быть представлена следующим образом:

Модифицированный метод

1. Выбрать начальное семейство $\mathcal{F}$ и угадать W^*.

2. Проверить совместность системы с дополнительным ограничением $W \ge W^*$.

3. Если система несовместна, то STOP, иначе найти $x(\mathcal{F})$.
4. Решить задачу Конкурента для $x(\mathcal{F})$, вычислить $LB(\mathcal{F})$ и $y(\mathcal{F})$.
5. Добавить $y(\mathcal{F})$ в семейство $\mathcal{F}$ и вернуться на шаг 2.

Заметим, что модифицированный метод также является конечным. Порождаемые решения $y(\mathcal{F})$ всегда добавляют существенные ограничения в задачу поиска W и не повторяются. Так как итерации оказываются менее трудоемкими, то возможности метода расширяются. Теперь удается решать задачи при $n = m = 100$ и $p = r = 5$. Более того, появляется возможность находить приближенные решения с гарантированной оценкой уклонения от оптимума. Для этого достаточно заменить вспомогательное ограничение $W \geq W^*$ на $W \geq W^*(1 + \Delta)$, где Δ — требуемая погрешность, и вместо W^* взять приближенное решение задачи.

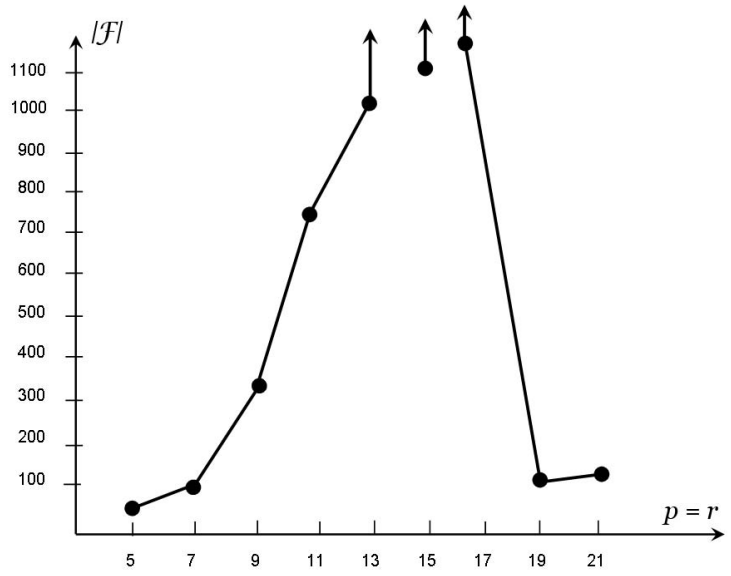


Рис. 23. *Зависимость размера семейства $\mathcal{F}$ от параметра p ($p = r$).*

На рис. 23 приведены результаты расчетов на указанных тестовых примерах. Целью расчетов было исследование зависимости размера семейства от параметров p и r. Эти параметры действительно сильно влияют на число итераций метода. При $n = 50, p = r = 17$ число итераций (размер семейства $\mathcal{F}$) превосходит 2000 и расчеты не укладываются в двое суток. При малых $p = r \leq 10$ задача не представляет большой трудности, хотя и является далеко не тривиальной, $|\mathcal{F}| \approx 700$. При $p = r \geq 20$ задача вырождается, так как почти во всех пунктах открывается производство либо Лидером, либо Конкурентом.

На рис. 24, 25 показана зависимость числа итераций при фиксирован-

ном p или r и изменении другого параметра.

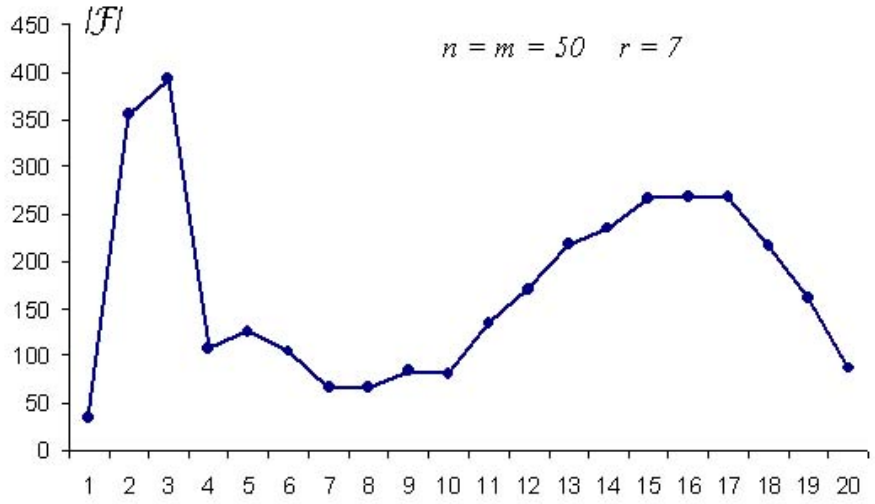


Рис. 24 *Зависимость размера семейства $\mathcal{F}$ от параметра p при фиксированном r ($r = 7$).*

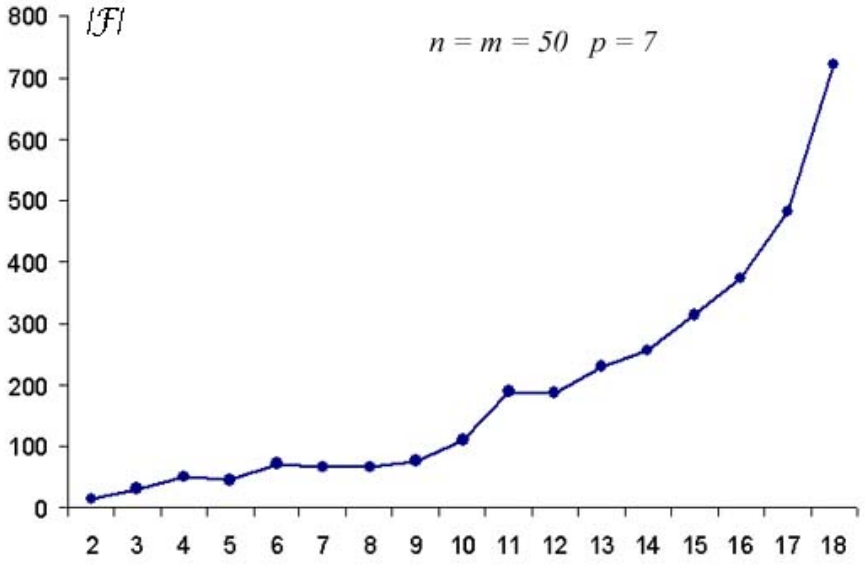


Рис. 25 *Зависимость размера семейства $\mathcal{F}$ от параметра r при фиксированном p ($p = 7$).*

В заключение заметим, что предложенный модифицированный метод является наглядным примером использования метаэвристик для точного решения задачи. На шаге 1 значение W^* проще всего находить именно таким способом. В частности, можно применять гибридный метод генетического локального поиска и вероятностного поиска с запретами из раздела 3.5.2. Конечно, такой подход не гарантирует нахождения оптимума W^*, но модифицированный метод будет работать и в этом случае,

если добавить еще один критерий остановки, а именно $W(\mathcal{F}) = LB(\mathcal{F})$. С таким дополнительным критерием метод будет находить оптимум W^* даже в том случае, когда на шаге 2 вместо W^* будет использоваться меньшая величина.

Дополнительные возможности совершенствования метода снова связаны с метаэвристиками. Заметим, что задача поиска допустимого решения допускает замену булевых переменных u_{jy} на непрерывные переменные $0 \le u_{jy} \le 1, j \in J, y \in \mathcal{F}$. Остается только одна группа булевых переменных $x_i \in \{0,1\}, i \in I$, по значениям которых однозначно восстанавливаются значения u_{jy}. Если $|\mathcal{F}| = 1$, то получаем аналог задачи Конкурента, то есть задачу о покрытии на максимум. Для задач такого рода успешно использовались метаэвристики. Более того, теперь для них есть естественный критерий остановки: $W \ge W^*$. Таким образом, дальнейшее сокращение трудоемкости одной итерации можно постараться достичь за счет методов локального поиска для задачи нахождения допустимого решения для Лидера. При этом последнюю итерацию, когда допустимого решения не существует для построенного семейства $\mathcal{F}$, по-прежнему придется решать методом ветвей и границ.

Заключение

В диссертации получены следующие основные результаты.

1. Разработаны и исследованы новые вероятностные методы локального поиска для решения NP-трудных задач о p-медиане, простейшей и многостадийной задач размещения, размещения с предпочтениями клиентов, задач стандартизации и унификации, а также для решения более сложной задачи о (r, p)-центроиде (конкурентной задачи о p-медиане), являющейся Σ_2^P–трудной. Эти итерационные методы позволяют находить оптимальные решения и в ряде случаев применяются для доказательства оптимальности получаемых решений.

2. Предложены и обоснованы итерационные методы вычисления апостериорных оценок точности получаемых приближенных решений для указанных выше труднорешаемых задач. В основе этих методов лежат оригинальные точные методы решения релаксированных задач.

3. Установлена вычислительная сложность нахождения локальных оптимумов для ряда задач размещения с полиномиально проверяемыми окрестностями. Показано, что эти задачи принадлежат классу PLS и являются наиболее трудными в этом классе. Получена экспоненциальная нижняя оценка на число итераций для алгоритмов локального улучшения при любом правиле выбора направления спуска. Доказана PSPACE-полнота задачи локального поиска при заданной стартовой точке.

4. Для задач размещения в игровой постановке установлена PLS-полнота задачи нахождения равновесных решений по Нэшу в чистых стратегиях. Получены достаточные условия эффективного вычисления приближенных равновесных решений. Показано, что в общем случае поиск приближенных равновесных решений является столь же сложной задачей, что и поиск самих равновесий по Нэшу.

5. Для экспериментального исследования численных методов и тестирования комплексов программ разработана электронная библиотека «Дискретные задачи размещения». В ней представлены тесты разной вычислительной сложности, оптимальные решения и результаты численных исследований для точных и итерационных методов локального поиска.

Литература

[1] Александров Д. А. Алгоритм муравьиной колонии для задачи о минимальном покрытии // Методы оптимизации и их приложения: Тр. 11-й Байкальской школы-семинара. — 1998. — Т. 3. — С. 17–20.

[2] Алексеева Е.В., Кочетов Ю.А. Генетический локальный поиск для задачи о p–медиане с предпочтениями клиентов. // Дискрет. анализ и исслед. операций. Сер. 2. — 2007. — Т. 14, № 1, — С. 3–31.

[3] Алексеева Е.В., Кочетов Ю.А., Кочетова Н.А. Критические параметры простейшей задачи размещения // Труды XIII Байкальской международной школы-семинара. — 2005. — Т.1. — С. 407-412

[4] Архипова Т.Т., Сергиенко И.В. О формализации и решении некоторых задач организации вычислительного процесса в системах обработки данных // Кибернетика. — 1973. — № 5.— С. 79–86.

[5] Архипова Т.Т., Сергиенко И.В. Метод возможных направлений и метод вектора спада в целочисленном программировании // Кибернетика. — 1975. — № 5. — С. 125–129.

[6] Береснев В.Л. Алгоритм неявного перебора для задачи типа размещения и стандартизации // Управляемые системы. / Сб. науч. тр. — Новосибирск: Ин-т математики СО АН СССР, 1974. — Вып. 12. — С. 24–34.

[7] Береснев В.Л. О задаче выбора оптимальных радов изделей и комплектующих узлов. I // Управляемые системы. / Сб. науч. тр. — Новосибирск: Ин-т математики СО АН СССР, 1977. — Вып. 16. — С. 35–46.

[8] Береснев В.Л. Алгоритмы минимизации полиномов от булевых переменных // Проблемы кибернетики. — М.: Наука, 1979. — Вып. 36. — С. 225–246.

[9] Береснев В. Л. Дискретные задачи размещения и полиномы от булевых переменных. — Новосибирск: Изд-во Инст. математики, 2005. — 408 с.

[10] Береснев В. Л., Гимади Э. Х., Дементьев В. Т. Экстремальные задачи стандартизации. — Новосибирск: Наука, 1978. — 335 с.

[11] Береснев В. Л., Гончаров Е. Н. Приближенный алгоритм для задачи минимизации полиномов от булевых переменных // Дискрет. анализ и исслед. операций. Сер. 2. — 1998. — Т. 5, № 2. — С. 3–19.

[12] Береснев В.Л., Ибрагимов Г.И., Кочетов Ю.А. Алгоритмы решения задачи оптимального выбора динамического ряда изделий // Управляемые системы / Сб. науч. тр. — Новосибирск: Ин-т математики СО АН СССР, 1984. — Вып. 24. — С. 3–19.

[13] Боровков А. А. Теория вероятностей. — М.: Наука, 1986. — 432 с.

[14] Васильев И.Л. Климентова К.Б. Кочетов Ю.А. Новые нижние оценки для задачи размещения с предпочтениями клиентов // Журнал вычисл. матем. и матем. физики. — 2009. — Т. 49, № 6. — С. 1055–1966.

[15] Васильев И.Л. Климентова К.Б. Плясунов А.В. Метод отсечений для двухуровневой задачи размещения // Труды XIV-й международной школы-семинара «Методы оптимизации и их приложения». — Иркутск, 2008. — Т. 1. — С. 577–585.

[16] Волконский В.А., Левина Л.В. Поманский А.Б. и др. Об одном итеративном методе решения задач целочисленного программирования // Доклады АН СССР. — 1966. — Т. 169, № 6. — С. 1289–1292

[17] Гимади Э. Х., Глебов Н. И., Дементьев В. Т. Об одном методе построения нижней нижней оценки и приближенного решения с апостериорной оценкой точности для задачи стандартизации // Управляемые системы: Сб. науч. тр. — Новосибирск: Ин-т математики СО АН СССР. — 1974. — Вып. 13. — С. 26-31.

[18] Глебов Н.И., Дементьев В.Т., Сычев А.Н. О динамике развития однородных технических систем // Управляемые системы / Сб. науч. тр. — Новосибирск: Ин-т математики СО АН СССР, 1971. — Вып. 8. — С. 51–67.

[19] Гончаров Е. Н. Метод ветвей и границ для простейшей двухуровневой задачи размещения предприятий // Дискрет. анализ и исслед. операций. Сер. 2. — 1998. — Т. 5, № 1. — С. 19–39.

[20] Гончаров Е. Н., Кочетов Ю. А. Вероятностный жадный алгоритм с ветвлением для многостадийной задачи размещения // Труды XI Байкальской школы-семинара «Методы оптимизации и их приложения». — Иркутск, 1998. — Т. 1. — С. 121–124.

[21] Гончаров Е. Н., Кочетов Ю. А. Поведение вероятностных жадных алгоритмов для многостадийной задачи размещения // Дискрет. анализ и исслед. операций. Сер. 2. — 1999. — Т. 6, № 1. — С. 12–32.

[22] Гончаров Е. Н., Кочетов Ю. А. Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискрет. анализ и исслед. операций. Сер. 2. — 2002. — Т. 9, № 2. — С. 13–30.

[23] Горбачевская Л. Е. Полиномиально разрешимые и NP-трудные задачи стандартизации: дис. ... канд. физ.-мат. наук. — Новосибирск: Ин-т математики им. С. Л. Соболева СО РАН. — 1998. — 123 с.

[24] Горбачевская Л. Е., Дементьев В. Т, Шамардин Ю. В. Двухуровневая задача стандартизации с условием единственности оптимального потребительского выбора // Дискрет. анализ и исслед. операций. Сер. 2. — 1999. — Т. 6, № 2. — С. 3–11.

[25] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.

[26] Емеличев В.А., Ковалев М.М., Кононенко А.М. Два приближенных метода решения задач целочисленного линейного программирования с булевыми переменными // — Автоматизированные системы управления. — Вып. 5. — Минск: ЦНИИТУ, 1971.

[27] Еремеев А. В. Генетический алгоритм для задачи о покрытии // Дискрет. анализ и исслед. операций. Сер. 2. — 2000. — Т. 7, № 1. — С. 47–60.

[28] Еремеев А. В. Разработка и анализ генетических и гибридных алгоритмов для решения задач дискретной оптимизации: дис. ... канд. физ.-мат. наук. — Омск: Омский филиал Ин-та математики им. С. Л. Соболева СО РАН. — 2000.

[29] Журавлев Ю.И. Локальные алгоритмы вычисления информации // Кибернетика. — 1965. — № 1. — С. 12–19.

[30] Ивахненко А. Г. Системы эвристической самоорганизации в технической кибернетике. — Киев: Техника, 1971.

[31] Корбут А.А., Сигал И.Х., Финкельштейн Ю.Ю. Гибридные методы в дискретном программировании// Изв. АН СССР. Техн. кибернет. — 1988. — № 1. — С. 65–77.

[32] Ковалев М.М., Пирьянович В.А. Случайный поиск локально-оптимальных планов задач дискретной оптимизации // Вопросы совершенствования планирования и управления народным хозяйством. Минск, НИИЭМП, 1975.

[33] Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ. — М.: МЦНМО, 1999. — 960 с.

[34] Кочетов Ю. А. Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения: Сб. лекций молодежных и научных школ по дискретной математике и ее приложениям. — М.: Изд-во центра прикл. исслед. при мех.-мат. фак. МГУ, 2001. — С. 84–117.

[35] Кочетов Ю.А. Двухуровневые задачи размещения // Труды ИВМ и МГ / Серия Информатика — Новосибирск: Изд. ИВМиМГ СО РАН, 2007. — Вып. 7. — С. 97–104.

[36] Кочетов Ю.А. Задача о (r, p)-центроиде // Материалы IV Всероссийской конференции Проблемы оптимизации и экономические приложения. — Омск: ОФ ИМ СО РАН, 2009. — С. 68–73.

[37] Кочетов Ю. А., Пащенко М. Г. Лагранжевы релаксации в задаче выбора оптимального состава системы технических средств // Управляемые системы: Сб. науч. тр. Новосибирск, 1993. — Вып. 31: Оптимизация дискретных структур. — С. 26–39.

[38] Кочетов Ю.А., Пащенко М.Г. Динамические задачи выбора оптимального состава системы технических средств // Дискрет. анализ и исслед. операций. Сер. 2. — 1995. — Т. 2, № 1. — С. 36–49.

[39] Кочетов Ю.А., Пащенко М.Г. Нижние границы в задаче выбора состава двухуровневой системы технических средств // Дискрет. анализ и исслед. операций. Сер. 2. — 1995. — Т. 2, № 4. — С. 32–41.

[40] Кочетов Ю.А., Пащенко М.Г., Плясунов А.В. О сложности локального поиска в задаче о p-медиане // Дискрет. анализ и исслед. операций. Сер. 2. — 2005. — Т. 12, № 2. — С. 44–71.

[41] Кочетов Ю.А., Плясунов А.В. Полиномиально разрешимый класс задач двухуровневого линейного программирования // Дискрет. анализ и исслед. операций. Сер. 2. — 1997. — Т. 4, № 2. — С. 23–33.

[42] Кочетов Ю.А., Плясунов А.В. Задача выбора ряда изделий с частичным внешним финансированием // Дискрет. анализ и исслед. операций. Сер. 2. — 2002. — Т. 9, № 2. — С. 78–96.

[43] Кочетов Ю.А., Столяр А.А. Использование чередующихся окрестностей для приближенного решения задачи календарного планирования с ограниченными ресурсами // Дискрет. анализ и исслед. операций. Сер. 2. — 2003. — Т. 10, №. 2. — С. 29–55.

[44] Крамер Г. Математические методы статистики. — М.: Мир, 1975. 468 с.

[45] Кротов Д.С. Нижние оценки числа m-квазигрупп порядка 4 и числа совершенных двоичных кодов// Дискрет. анализ и исслед. операций. Сер. 1. — 2000. — Т. 7, №. 2. — С. 47–53//

[46] Лбов Г.С. Выбор эффективной системы зависимых признаков // Вычислительные системы. — 1965. — Вып. 19. — С. 21–34.

[47] Лебедев С.С., Ковалевская М.И. Множители Лагранжа в простейшей задаче размещения // Исследования по дискретной оптимизации. — М.: Наука, 1976. — С. 170–180.

[48] Михалевич В. С., Трубин В. А., Шор Н. З. Оптимизационные задачи производственно-транспортного планирования. — М.: Наука, 1986. 264 с.

[49] Мулен Э. Теория игр с примерами из математической экономики. — М.: Мир, 1985. — 200 с.

[50] Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. — М.: Мир, 1985.

[51] Пащенко М.Г. Лагранжевы релаксации в динамических задачах выбора оптимального состава системы технических средств. Дис. ... канд. физ.-мат. наук. — Новосибирск: Ин-т математики им. С. Л. Соболева СО РАН. — 1998. — 79 с.

[52] Пащенко М.Г. Применение Лагранжевых релаксаций в локальном поиске для обобщенной задачи о назначении // Труды 12-й Байкальской международной конференции "Методы Оптимизации и их Приложения", 2001, Т. 1, — С. 180–185.

[53] Растригин Л. А. Статистические методы поиска. — М.: Наука, 1968.

[54] Растригин Л. А. Случайный поиск — специфика, этапы истории и предрассудки // Вопросы кибернетики. — 1978. — Вып. 33. — С. 3–16.

[55] Сергиенко И.В. Применение метода вектора спада для решения задач целочисленного программирования // Управляющие системы и машины. — 1975. — № 3.

[56] Сергиенко И.В., Каспшицкая М.Ф. Модели и методы решения на ЭВМ комбинаторных задач оптимизации. — Киев: Наук.думка, 1981.

[57] Сигаев В. С. О свойствах локальных оптимумов задачи размещения буферных накопителей // Вестник Омского университета. — 2006. — № 4. — С. 1–3.

[58] Сигал И.Х. Последовательность применения алгоритмов приближенного решения в комбинированном алгоритме решения задачи коммивояжера // Журн. вычисл. матем. и матем. физики. — 1989. — Т. 29, № 11. — С. 1714–1721.

[59] Сигал И.Х., Иванова А.П. Введение в прикладное дискретное программирование. — М: Физматлит, 2007. 304 с.

[60] Трубин В.А. О методе решения задач целочисленного линейного программирования специального вида // Доклады АН СССР. — 1969. — Т. 189, № 5.

[61] Трубин В.А., Чумаков Б.М. Задачи унификации и размещения технических средств // Теория оптимальных решений. — Киев: ИК АН УССР, 1979. — С. 69–75.

[62] Финкельштейн Ю.Ю. Приближенные методы и прикладные задачи дискретного программирования. — М.: Наука, 1976.

[63] Финкельштейн Ю.Ю. Об одном классе задач дискретного программирования // Эконом. и матем. методы. — 1968. — Т. 4, № 4. — С. 652–655.

[64] Финкельштейн Ю.Ю. О решении задач дискретного программирования специального вида // Эконом. и матем. методы. — 1965. — Т. 1, № 2. — С. 262–270.

[65] Фогель Л., Оуэнс А., Уолш М. Искусственный интеллект и эволюционное моделирование. — М.: Мир, 1969.

[66] Форд Л., Фалкерсон Д. Потоки в сетях. — М.: Мир, 1966.

[67] Хачатуров В.Р., Астахов Н.Д. Динамические задачи размещения (модели и методы решения)// Экономика и математические методы. — 1976. — Т. 12, Вып. 1. — С. 93–109.

[68] Хачатуров В. Р., Веселовский В. Е., Злотов А. В. и др. Комбинаторные методы и алгоритмы решения задач дискретной оптимизации большой размерности. — М.: Наука, 2000.

[69] Черенин В.П., Хачатуров В.Р. Решение методом последовательных расчетов одного класса задач о размещении производства // Экономико–математические методы. М.: Наука, 1965. — Вып. 2. — С. 279–290.

[70] Aarts E. H. L., Korst J. H. M., Laarhoven P. J. M. Simulated annealing // Local search in combinatorial optimization. Chichester: John Wiley & Sons, 1997. — P. 91–120.

[71] Aarts E. H. L., Lenstra J. K. (Eds.) Local Search in Combinatorial Optimization. — Chichester: John Wiley & Sons, 1997.

[72] Aggarwal C.C., Orlin J.B., Tai R.P. Optimized crossover for maximum independent set // Oper. Res. — 1997. — V. 45. — P. 225–234.

[73] Ahuja R.K., Ergun O., Orlin J.B., Punnen A.P. A survey of very large–scale neighborhood search techniques // Discrete Appl. Math. — 2002. — V. 123, Issue 1–3. — P. 75–102.

[74] Ahuja R.K., Orlin J. A milti-exchange heuristic for the single-source capacitated facility location problem. // Manag. Sci. — 2004. — V. 50, — C. 749–760.

[75] Alekseeva E., Kochetov Yu., Plyasunov A. Complexity of local search for the p-median problem // European J. Oper. Res. — 2008. — V. 191 — P.736-752.

[76] Alexandrov D., Kochetov Y. Simple plant location problem with partial external finance: lower bound, heuristic and exact solution // Operations Research Proceedings 1996 — Berlin: Springer, 1997. — C. 90–94.

[77] Anderson E.J., Glass C.A., Potts C.N. Machine scheduling// In: E.Aarts, J.K. Lenstra Local Search in Combinatorial Optimization. — Chichester: John Wiley & Sons, 1997, — P. 361–414.

[78] Angel E., Zissimopoulos V. On the classification of NP-complete problems in terms of their correlation coefficient // Discrete Appl. Math. — 2000. — V. 99. — P. 261–277.

[79] Arora S., Raghavan P., Rao S. Approximation schemes for Euclidean k-medians and related problems // Proc. 30th Annual ACM Symposium on Theory of Computing. — 1998. — P. 106–113.

[80] Ausiello G., Crescenzi P., Gambosi G., Kann V., Marchetti-Spaccamela A., Protasi M. Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties. — Berlin: Springer–Verlag, 1999.

[81] Ausiello G., Protasi M. Local search, reducibility and approximability of NP-optimization problems// Information Processing Letters. — 1995. — V. 54. — P. 73–79.

[82] Avella P., Boccia M., Martino C.D., Oliviero G., Sforza A., Vasil'ev I. A decomposition approach for a very large scale optimal diversity management problems // 4OR. — 2005. — V. 3. — P. 23–37.

[83] Avella P., Sassano A., Vasil'ev I. Computational study of large-scale p-median problems // Math. Program. Ser. A. — 2007. — V. 109. — P. 89–114.

[84] Bahiense, L., N. Maculan, and C. Sagastizábal. The volume algorithm revisited: relation with bundle methods // Math. Prog. — 2002. — V. 94, N 1. — P. 41–70.

[85] Balas E., Niehaus W. Finding large cliques in arbitrary graphs by bipartite matching // D.S. Johnson and M.A. Trick (eds.) Cliques, coloring, and satisfiability. DIMACS Ser. Discrete Math. Theoret. Comput. Sci. — 1996. — V. 26. — P. 29–49.

[86] Balas E., Niehaus W. Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems // J. Heuristics. — 1998. — V. 4, N 4, — P. 107–122.

[87] Barahona, F. and R. Anbil. The Volume Algorithm: Producing Primal Solutions with a Subgradient Method // Math. Prog. — 2000. — V. 87, N.3. — P. 385–400.

[88] Barros A.I., Labbe M. A general model for the uncapacitated facility and depot location problem // Location Science. — 1994. — V. 2, N 3. — P. 173–191.

[89] Battiti R., Protasi M. Reactive local search for maximum clique // Proc. of Workshop on Algorithm Engineering, 1997. — P. 74–82.

[90] Benati S., Laporte G. Tabu search algorithms for the $(r|X_p)$-medianoid and $(r|p)$-centroid problems. // Location Science.— 1994. — V.2, N.2, — P. 193–204.

[91] Bhadury J., Eiselt H.A., Jaramillo J.H. An alternating heuristic for medianoid and centroid problems in the plane. // Computers and Operations Research. — 2003. — V.30. — P. 553–565.

[92] Bilde O., Krarup J. Sharp lower bounds and efficient algorithms for the simple plant location problem // Annals Discrete Math. 1. — Amsterdam: North Holland Publ. — 1977. — P. 79–97.

[93] Blum C. Roli A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison // ACM Computing Surveys (CSUR). — 2003. — V.35, Issue 3. — P. 268–308.

[94] Boese K. D., Kahng A. B., Muddu S. A new adaptive multi-start technique for combinatorial global optimizations // Oper. Res. Lett. — 1994. — V. 16, N 2. — P. 101–113.

[95] Bock F. An algorithm for solving "travelling-salesman"and related network optimization problems // Bulletin Fourteenth National Meeting of the Operations Research Society of America. — 1958. — P. 897.

[96] Boros E., Hammer P. L. Pseudo-Boolean optimization // Discrete. Appl. Math. — 2002. — V. 123. — P. 155–225.

[97] Bremermann H. J., Roghson J., Salaff S. Global properties of evolution processes. In Natural automata and useful simulations. Edited by Pattee H. H. etc. — London: Macmillan, 1966. — P. 3–42.

[98] Burke E.K., Kendall G. (Eds.) Search Methodologies. Introductory Tutorials // Optimization and Decision Support Techniques. — Berlin: Springer, 2005.

[99] Charon I. Hudry H. The noising methods. A survey. In: Ribeiro C.C. Hansen P. (eds.) Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization. — Boston: Kluwer. Acad. Publ., 1998. — P. 245–262.

[100] Christofides N. Beasley R. Extensions to a Lagrangean relaxation approach for the capacitated warehouse location problem // European Journal of Operational Research — 1984. — V.12 — P. 19–28.

[101] Cornuejols G. Sridharan R. Thizy J.M. A comparison of heuristics and relaxation for the capacitated plant location problem // European Journal of Operational Research — 1991. — V. 50 — P. 280–297.

[102] Croes G. A. A method for solving traveling salesman problems // Oper. Res. — 1958. — V. 6. — P. 791–812.

[103] Dobson G., Karmarkar U. Competitive location on a network. // Oper. Res. — 1987, — V. 35. — P. 565–574.

[104] Dolgui A., Eremeev A., Kolokolov A., Sigaev V. A Genetic Algorithm for the Allocation of Buffer Storage Capacities in a Production Line with Unreliable Machines // Journal of Mathematical Modelling and Algorithms. — 2002. — V.1, N 2. — P. 89-104.

[105] Dreo J., Petrowski A., Siarry P., Taillard E., Metaheuristics for Hard Optimization, — Springer, 2006.

[106] Drezner Z. (Ed.) Facility Location. A Survey of Applications and Methods. — Springer, 1995. — 571 p.

[107] Drenzer Z., Eiselt H.A. Consumers in competitive location models // Z. Drenzer, H. Hamacher (Eds.) Facility Location. Applicatios and Theory. — Springer, 2004. — P. 151–178.

[108] Drezner Z., Klamroth K., Schobel A., Wesolowsky G. The Weber Problem // Z. Drezner, H. Hamacher (Eds.) Facility Location. Applications and Theory. — Springer, 2004. — P. 1–36.

[109] Dyer M.E. A O(n) algorithm for the multiple–choice knapsack linear program // Math. Programming. — 1984. — V.29, N.1. — P. 57–63.

[110] Everett H. Generalized Lagrange multiplies method for solving problems of optimum allocation of resources // Oper. Res. — 1963. — V. 11 — P. 399–417.

[111] Eiselt H. A., Sandblom C. -L. Decision Analysis, Location Models, and Scheduling Problems. — Springer, 2004. — 380 p.

[112] Erlenkotter D. A dual-based procedure for uncapacitated facility location // Oper. Res. — 1978. — V. 26. — P. 992–1009.

[113] Erlenkotter D. A comparative study of approaches to dynamic location problems // European J. Oper. Res. — 1981. — V. 6, N 2. — P. 36–81.

[114] Fabrikant, A., Papadimitriou C., Talwar K. The complexity of pure Nash equilibria // Proc. 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, 2004. — P. 604-612.

[115] Faigle U., Kern W. Some convergence results for probabilistic tabu search // ORSA J. Comput. — 1992. — V. 4, N 4. — P. 32–37.

[116] Fathian M., Amiri B., Maroosi A. Application of honey-bee mating optimization algorithm on clustering // Applied Mathematics and Computation. — 2007. — V. 190(2). — P. 1502-1513.

[117] Feige U. A threshold of $\ln n$ for approximating set cover // Proceedings of the 28 annual ACM symposium on the theory of computing. — N.Y.: ACM Press, 1996. — P. 314–318.

[118] Feo T. A. Greedy randomized adaptive search procedures // J. Global Optim. — 1995. — V. 6. — P. 109–133.

[119] Festa P., Resende M. G. C., GRASP: An annotated bibliography. In: C. C. Ribeiro, P. Hansen (Eds.) Essays and surveys in metaheuristics. — Kluwer Academic Publishers, 2002. — P. 325–368.

[120] Fiduccia C.M., Mattheyses R.M. A linear-time heuristic for improving network partitions // Proc. 19th Design Automation Conference. Los Alamitos, CA: IEEE Comput. Soc. Press, 1982. — P. 175–181.

[121] Fischetti M., Lodi A. Local branching. // Math. Programming. — 2003. — V. 98, N. 2. — P. 23–47.

[122] Fisher M. L. Worst-case analysis of heuristic algorithms // Manag. Sci. — 1980. — V. 26, N. 1. — P. 1–18.

[123] Fisher M., Kedia P. Optimal solution of set covering partitiong problems using dual heuristics // Management Science. — 1990. — V. 36, N.6. — P.674–688.

[124] Fischer S. T. A note on the complexity of local search problems // Information Processing Letters. — 1995. — V. 53. — P. 69–75.

[125] Frangioni A. About Lagrangian methods in integer optimization // Annals of Operations Research. — 2005. — V. 139. — P. 163–193.

[126] Fridman M. L., Johnson D. S., McGeoch L. A., Ostheimer G. Data structures for traveling salesman // J. Algorithms. — 1995. — V. 18. — P. 432–479.

[127] Geoffrion, A.M. (1974). Lagrangian Relaxation and its Uses in Integer Programming // Math. Prog. Study. — 1974. — V. 2. — P. 82–114.

[128] Glover F. Future paths for integer programming and links to artificial intelligence. // Comp. Oper. Res. — 1986. — V. 13. — P. 533–549.

[129] Glover F., Laguna M. Tabu Search. — Dordrecht: Kluwer Acad. Publ., 1997.

[130] Goldberg D. E. Simple genetic algorithms and the minimal deceptive problem. Genetic Algorithms and Simulated Annealing. Chapter 6. — Los Altos, CA, Morgan Kauffman, 1987. — P. 74–88.

[131] Goldberg D. E. Genetic algorithm in search, optimization and machine learning. — Reading, MA: Addison-Wesley. 1989.

[132] Guha S., Khuller S. Greedy strikes back: improved facility location algorithms. // J. Algorithms. — 1999. — V. 31. — P. 228–248.

[133] Guignard, M. Lagrangean Relaxation // TOP. — 2003. — V. 11, N.2. — P. 151–228.

[134] Guignard M., Kim S. Lagrangian decomposition: a model yelding stronger Lagrangian bounds // Math. Prog. — 1987. — V. 39. — P. 215–228.

[135] Gusfield D., Martel C., Fernandez-Baca D. Fast algorithms for bipartite network flow // SIAM J. Comput. — 1987. — V. 16, N 2. — P. 237–251.

[136] Gutin G. Exponential neighborhood local search for the traveling salesman problem // Comput. Oper. Res. — 1999. — V. 26. — P. 313–320.

[137] Gutin G., Yeo A. Small diameter neighborhood graphs for the traveling salesman problem: four moves from tour to tour // Comput. Oper. Res. — 1999. — V. 26. — P. 321–327.

[138] Hakimi S. L. On locating new facilities in a competitive environment // European J. Oper. Res. — 1983. — V. 12, N 1. — P. 29–35.

[139] Hakimi S. L. Locations with spatial interactions: competitive locations and games // P. B. Mirchandani, R. L. Francis (Eds.) Discrete Location Theory. — Wiley & Sons, 1990. — P. 439–478.

[140] Hall M.Jr. Combinatorial Theory. — Blaisdell. Waltham. MA, 1967.

[141] Hammer P.L., Rudeanu S. Boolean method in operations research and related areas. — Berlin: Springer-Verlag, 1968.

[142] Hanjoul P., Peeters D. A facility location problem with clients' preference orderings // Regional Science and Urban Economics. — 1987. — V. 17. — P. 451–473.

[143] Hansen, P., Brimberg, J., Urosević, D., Mladenović, N. Primal-Dual Variable Neighbourhood for the Simple Plant Location Problem // INFORMS J. Computing. — 2007. — V. 19, N 4. — P. 552–564.

[144] Hansen P., Kochetov Yu., Mladenović N. Lower bounds for the uncapacitated facility // Discrete Optimization Methods in Production and Logistics (DOM'2004) / 2^{nd} International Workshop; proceedings. — Omsk: Nasledie Dialog-Sibir., 2004. — P. 50–55.

[145] Held, M., R. Carp. The travelling salesman problem and minimum spanning trees: Part II // Math. Prog. — 1971. — V. 1. — P. 6–25.

[146] Held M., Wolfe P., Crowder H. P. Validation of subgradient optimization // Math. Programming. — 1974. — V. 6, N 1. — P. 62–88.

[147] Hertz A., Plumettaz M., Zufferey N. Variable Space Search for Graph Coloring // Discrete Appl. Math. — 2008. — V. 156. — P. 2551–2560.

[148] Hertz A., Taillard E., de Werra D. Tabu search // Local search in combinatorial optimization. — Chichester: John Wiley & Sons, 1997. — P. 121–136.

[149] Holland J. H. Adaptation in Natural and Artificial Systems. — Ann Arbor: University of Michigan Press, 1975.

[150] Ibaraki T., Nonobe K., Yagiura M. (Eds.) Metaheuristics: progress as real solvers. — Berlin: Springer, 2005.

[151] Jacobsen S.K. Heuristic solutions to dynamic plant location problems // Advances in Operations Research: Proc. EURO II / The second European Congress on Operations Research. — Amsterdam, 1977. — P. 207–213.

[152] Johnson D.S. Local optimization and the traveling salesman problem // Automata, Languages, and Programming. — Berlin: Springer Verlag, 1990. (Lecture Notes in Computer Science, V. 443) — P. 446–461.

[153] Johnson D.S., McGeoch L.A. The traveling salesman problem: A case study. // In: E.H.L. Aarts, J.K. Lenstra (eds) *Local Search in Combinatorial Optimization.* — Chichester: John Wiley & Sons, 1997. — P. 215–310.

[154] Johnson D.S., Papadimitriou C.H., Yannakakis M. How easy is local search? // J. of Computer and System Sciences. — 1988. — V. 37. — P. 79–100.

[155] Kellerer H., Pferschy U., Pisinger D. Knapsack Problems. — Berlin: Springer. – 2004. — 546 p.

[156] Kernighan B.W., Lin S. An efficient heuristic procedure for partitioning graphs // Bell System Technical Journal. — 1970. — V. 49. — P. 291–307.

[157] Kochetov Y., Alekseeva E., Levanova T., Loresh M. Large neighborhood local search for the p-median problem // Yugoslav Journal of Oper. Res. — 2005. — V. 15, № 1. — P. 53–63.

[158] Kochetov Yu., Ivanenko D. Computationally difficult instances for the uncapacitated facility location problem. In: T. Ibaraki et al. (eds.) Metaheuristics: progress as real solvers. — Springer, 2005, — P. 351-367.

[159] Kochetov Yu., Kononova P., Paschenko M. Formulation Space Search Approach for the Teacher/Class Timetabling Problem // Yugoslav Journal of Operations Research. — 2008, — V. 18, N 1. — P. 1–11.

[160] Korte B., Vygen J. Combinatorial Optimization. Theory and Algorithms / Third Edition. — Springer, 2005. — 588 p.

[161] Koutsoupias E., Papadimitriou C. Worst-case equilibria // Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science. — Trier: Germany, 1999. — P. 404–413.

[162] Koza J. R. Genetic Programming II. Automatic Discovery of Reusable Programs (Complex Adaptive Systems). — MIT Press, 1994.

[163] Krarup J., Pruzan P. M. The simple plant location problem: survey and synthesis. // European J. Oper. Res. — 1983. — V. 12, N 1. — P. 36–81.

[164] Krasnogor N., Hart W., Smith J. (eds) Resent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing. V. 166. — Berlin: Springer, 2004

[165] Krentel M.W. On finding and verifying locally optimal solutions // SIAM J. on Comput. — 1990. — V.19. — P. 742–751.

[166] Lemaréchal, C. . Lagrangian Relaxation. In: M. Jünger and D. Naddef (eds.), Computational Combinatorial Optimization, — Springer Verlag, Heidelberg, 2001. — P. 115–160.

[167] Martello S., Toth P. Knapsack Problems. Algorithms and Computer Implementations. — Chichester: John Wiley & Sons, — 1990. — 296 p.

[168] Mautor T. Intensification neighborhoods for local search methods // Essays and surveys in metaheuristics. — Boston: Kluwer Academic Publishers, 2002. — P. 493–508.

[169] Mirchandani P. B., Francis R. L.(Eds.) Discrete Location Theory. — Wiley, 1990. — 576 p.

[170] Mladenović N.,Brimberg J.,Hansen P., Moreno-Pérez J. The p-median problem: A survey of metaheuristic approaches // European J. of Oper. Res. — 2007. — V. 179, N 3. — P. 927–939.

[171] Mladenovic N , Brimberg J, Hansen P and Moreno-Perez J. The p-median problem: A survey of metaheuristic approaches. // European J Operational Research. — 2007, — V. 179. — P. 927–939.

[172] Mladenović, N., Plastria, F., Urošević, D. Reformulation descent applied to circle packing problems. // Computers and Operations Research. — 2005. — V. 32, N. 9. — P. 2419–2434.

[173] Moscato P. Memetic algorithms: a shot introduction. // In: D. Corne, M. Dorigo, F. Glover (eds.) *New Ideas in Optimization.* — London: McGraw–Hill, — 1999. — P. 219–234.

[174] Moscato P., Cotta C. A gentle introduction to memetic algorithms. In: F. Glover, G. Kochenberger (eds) Handbook of Metaheuristics. — Norwell, MA: Kluwer Acad. Publ., 2003.

[175] Mühlenbein H. Parallel genetic algorithm, population dynamics and combinatorial optimization // Proc. Third Inter. Conf. Genetic Alg. — San Mateo: Morgan Kaufman, 1989. — P. 416–421.

[176] Nicholson T.A.J. A sequential method for discrete optimization problems and its application to the assignment, traveling salesman and tree scheduling problems // J. Inst. Math. Appl. — 1965. — V. 13. — P. 362–375.

[177] Noltermeier H., Spoerhose J., Wirth H.C. Muliple voting location and single voting location on trees. // European J. Oper. Res.— 2007. — V. 181. — P. 654–667.

[178] Orlin J.B., Punnen A.P., Schulz A. Approximate local search in combinatorial optimization // SIAM J. Comput. — 2004. — V. 33. — P. 1201–1214.

[179] Osman I.H., Laporte G. Metaheuristics: a bibliography // Ann. Oper. Res. — 1996. — V. 63. — P. 513–628.

[180] Papadimitriou C.H. The complexity of the Lin-Kernighan heuristic for the traveling salesman problem // SIAM J. on Comput. — 1992. — V. 21. — P. 450–463.

[181] Papadimitriou C. H. Computational complexity. — Addison–Wesley, 1994.

[182] Plastria F., Vanhaverbeke L. Discrete models for competitive location with foresight // Computers and Operations Research. — 2008. — V. 35, N 3. — P. 683–700.

[183] Poljak, B.T. Subgradient Methods: A Survey of Soviet Research. In: C. Lemaréchal and R. Mifflin (eds.), Nonsmooth Optimization, IIASA Proceedings Series, 1977. — vol. 3.

[184] Rechenberg I. Evolutionsstrategie: Optimerung Technischer Systeme nach Prinzipen der Biologischen Evolution. — Stuttgart: Formann–Holzboog Verlag, 1973.

[185] Resende M. G. C., Werneck P. F. A hybrid heuristic for the p-median problem // Journal of Heuristics. — 2004. — V. 10. — P. 59–88.

[186] Resende M. G. C., Werneck P. F. A hybrid multistart heuristic for the uncapacitated facility location problem // European Journal of Oper. Res. — 2006. — V. 174, — P. 54–68.

[187] Reeves C. R. Genetic algorithms for the operations researcher // INFORMS J. Comput. — 1997. — V. 9, N 3. — P. 231–250.

[188] Rhys J.M.W. A selection problem of shared fixed costs and network flows // Manag. Sci. — 1970. V. 17. — P. 200–207.

[189] Ribeiro C.C. Hansen P. (Eds.) Meta-heuristics: advances and trends in local search paradigms for optimization. Boston: Kluwer. Acad. Publ., 1998.

[190] Rodriguez C. M. C., Perez J. A. M. Multiple voting location problems // European J. Oper. Res. – 2008. — V. 191, N 2. — P. 437–453.

[191] Rudolph G. Finite Markov chain results in evolutionary computation: a tour d'horizon // Fundamenta Informaticae. — 1998. — V. 35, N 1-4, — P. 67–89.

[192] Shor N.Z. Minimization methods for non-differentiable functions. — Berlin: Springer, 1985.

[193] Sastry K., Goldberg D., Kendall G. Genetic algorithms. // In: E.K. Burke, G. Kendall (Eds.) *Search Methodologies. Introductory Tutorials in Optimization and Decision Support Techniques.* — New York: Springer, 2005. — P. 97–126.

[194] Schrijver A. Combinatorial optimization. Polyhedra and efficiency. — Berlin. Springer, 2003. — 1881 p.

[195] Schäffer A. A., Yannakakis M. Simple local search problems that are hard to solve // SIAM J. on Comput. — 1991. — V. 20, N. 1. — P. 56–87.

[196] Schuler R., Schöning U., Watanabe O. An improved randomized algorithm for 3-SAT // Techn. Rep., Dept. of Math. and Comput. Sci. Tokyo Inst. of Technology, 2001.

[197] Schumacher C. Black box search — framework and methods. Ph. D. Thesis. — The Univ. of Tennessee, Knoxville, USA, 2000.

[198] Schwefel H. P. Numerical optimization of computer models. — Chichester: Wiley, 1981.

[199] Spoerhose J., Wirth H.C. (r; p)-Centroid problems on paths and trees. Tech. Report 441, — Inst. Comp. Science, University of Würzburg, — 2008.

[200] Stadler P.F. Corelation in landscapes of combinatorial optimization problems // Europhys. Lett. — 1992. — V. 20. — P. 479–482.

[201] Stadler P.F., Schnabl W. The landscape of the traveling salesman problem // Physics Letters A. — 1992. — V. 161. — P. 337–344.

[202] Talbi E.G. A taxonomy of hybrid metaheuristics // Journal of Heuristics. — 2002, — V. 8. — P. 541–564.

[203] Tcha D. W., Lee B.-Y. A branch-and-bound algorithm for the multilevel uncapacitated facility location problem. // European J. Oper. Res. — 1984. — V. 18, N 1. — P. 35–43.

[204] Van Roy T., Erlenkotter D. A dual-based procedure for dynamic facility location // Management Sci. — 1982. — V. 28, N 10. — P. 1091–1105.

[205] Verhoeven M.G.A., Swinkels P.C.J., Aarts E.H.L. Parallel local search and the traveling salesman problem // Working paper, Philips research laboratories. — Eindhoven, 1995.

[206] Vetta A. Nash equilibria in competitive societies, with applications to facility location, traffic routing and auctions // Proceedings of 43-rd Annual IEEE symposiun on foundations of computer science. — Vancouver: Canada, 2002. — P. 416–425.

[207] Weiner P., Savage S.L., Bagchi A. Neighborhood search algorithms for guaranteeing optimal traveling salesman tours must be inefficient // J. of Computer and System Sciences. — 1976. — V.12. — P.25–35.

[208] Wilbaut C., Hanafi S., Freville A., Balev S. Tabu search: global intensification using dynamic programming. // Control and Cybernetics. — 2006. — V. 35, N. 3. — P. 579–588.

[209] Wolpert D. H., Macready W. G. No free lunch theorem for search // Techn. Rep. SFI-TR-95-02-010. — Santa Fe Inst., 1995.

[210] Vredeveld T., Lenstra J.K. On local search for the generalized graph coloring problem // Oper. Res. Letters. — 2003. — V. 31. — P. 28–34.

[211] Yagiura M., Ibaraki T., Glover F. An ejection chain approach for the generalized assignment problem // INFORMS Journal on Computing. — 2004, — V. 16, N. 2. — P. 133–151.

[212] Yannakakis M. Computational complexity // Local search in combinatorial optimization. — Chichester: Wiley, 1997. — P. 19–55.