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

На правах рукописи
УДК 519.21

Алексеева Екатерина Вячеславовна

Алгоритмы локального поиска для задачи о p-медиане с предпочтениями клиентов

01.01.09 — дискретная математика и математическая кибернетика

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научные руководители:
д.ф.-м.н., профессор
В.Л. Береснев
к.ф.-м.н., доцент
Ю.А. Кочетов

Новосибирск — 2007

ОГЛАВЛЕНИЕ

Введение

Теория дискретных задач размещения является одной из интенсивно развивающихся областей в исследовании операций. Возникновение этих задач и первые попытки их решения приписывают французским и итальянским математикам 17 века и, в частности, Пьеру Ферма (1601–1665). Он интересовался следующей задачей. Заданы три точки на плоскости. Найти такую четвертую точку, что сумма расстояний от неё до трех заданных точек была бы минимальной. Решением этой задачи занимался ученик Галилея, итальянский математик Евангелиста Торричелли (1608–1647). Возможно, что первым, кто сформулировал и решил эту задачу был итальянский математик Батисто Ковальери (1598–1647), однако точное первенство установить уже очень трудно. В начале двадцатого века (1909) Альфред Вебер исследовал более общую задачу о нахождении центра тяжести для трех взвешенных точек, а позже (1941) Курант и Роббинс популяризировали так называемую задачу Штейнера (1796–1863) о нахождении кратчайшей остовной сети для трёх точек на плоскости.

По-настоящему бурное развитие модели размещения получили с рождением вычислительной техники. Линейные модели, модели частично–целочисленного программирования, статистические и динамические модели размещения, а также модели с нелинейными целевыми функциями рождались из приложений по размещению нефтяных и газовых станций, размещению предприятий, станций метро, милицейских и пожарных участков и др. В СССР первые модели размещения предприятий исследовались В.П. Черениным и В.Р. Хачатуровым [19]. Интерес к этим моделям в Институте математике им. С.Л. Соболева СО РАН связан в первую очередь с приложениями в области стандартизации и унификации [1, 17, 18]. Работы В.Л. Береснева, Э.Х. Гимади, В.Т. Дементьева, Н.И. Глебова, а позже А.И. Давыдова, Ю.А. Кочетова, А.В. Плясунова, Ю.В. Шамардина, Л.Е. Горбачевской, А.А. Агеева и др. создали фундамент для разработки программно математического аппарата решения соответствующих экстремальных задач.

В общем виде дискретные задачи размещения моделируют ситуации, в которых из заданного множества объектов требуется выбрать некоторое подмножество так, чтобы удовлетворить заданные потребности с наименьшими затратами или наибольшей прибылью. В качестве объектов могут выступать филиалы крупной компании, складские комплексы, компоненты микросхем и многое другое.

Дискретные задачи размещения имеют конечную область допустимых решений, поэтому заранее известно, что в задаче существует оптимальное решение. При небольшой размерности, оптимальное решение можно найти, перебрав все множество допустимых решений. Однако реальные размерности задач ограничивают возможности переборного алгоритма даже с применением современных вычислительных технологий. Поэтому основная цель, которую преследует большинство ученых в этой области заключается в разработке непереборных алгоритмов, способных за разумное время найти оптимальное или близкое к оптимальному решение задачи. Известно, что многие задачи размещения относятся к классу NP-трудных задач, т. е. построение точных полиномиальных алгоритмов скорее всего является бесперспективным подходом. Поэтому исследования дискретных задач размещения и разработка алгоритмов их решения ещё долгое время будет актуальной темой.

Основными направлениями исследований, как правило, являются:
– выявление полиномиально разрешимых случаев задачи, построение для них полиномиальных алгоритмов;
– разработка приближённых алгоритмов с априорными оценками точности;
– разработка итерационных методов локального поиска и эвристических алгоритмов.
В данной работе исследования ведутся по последнему направлению.

Существует большое число прикладных задач, для которых соответствующие математические модели принадлежат классу дискретных задач размещения. Основополагающими задачами в этом классе принято считать: простейшую задачу размещения, задачу о p-медиане и задачу о p-

центре. Несмотря на то, что эти задачи являются хорошо изученными и им посвящено огромное число публикаций, они до сих пор привлекают внимание ученых, т. к. составляют основу других интересных моделей и аккумулируют в себе основные математические трудности.

Так как диссертационная работа посвящена одному обобщению задачи о p-медиане, то приведём сначала математическую формулировку классической задачи о p-медиане и дадим краткий обзор основных известных результатов.

Пусть задано два конечных множества $I = \{1, \dots, m\}$ и $J = \{1, \dots, n\}$. Первое множество будем интерпретировать как множество возможных пунктов размещения предприятий для производства некоторого однородного продукта, второе — множество клиентов, использующих этот продукт. Известны величины $c_{ij}, \quad i \in I, \quad j \in J$, задающие транспортные затраты на обслуживание j-го клиента из i-го пункта. Требуется выбрать из множества I не более p пунктов, в которых следует разместить предприятия, т. е. найти подмножество $S \subset I$ мощности не более p так, чтобы суммарные затраты на обслуживание всех клиентов были бы минимальными. Комбинаторная формулировка задачи имеет следующий вид:

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

Установлено [48], что задача о p-медиане является NP-трудной. В [5] показано, что если матрица (c_{ij}) есть матрица кратчайших расстояний на дереве, то задача полиномиально разрешима. В [61] доказано, что в общем случае существование приближённого алгоритма с гарантированной относительной погрешностью не более $2^{q(n,m)}$, где q — произвольный полином от размерности задачи, влечёт совпадение классов P и NP. Особое внимание в литературе уделяется метрическому случаю, когда элементы матрицы (c_{ij}) удовлетворяют неравенству треугольника. В этом случае стандартный алгоритм локального спуска приводит к локальному оптимуму с относительной погрешностью не более $(3 + \frac{2}{k})$, где k — число предприятий, которые могут заменяться в решении на другие при каждом локальном улучшении [25]. Каждый шаг такой итерационной процедуры имеет трудоемкость не более $O(m^k)$, но число шагов может оказаться экспоненциальным. Лучший из известных полиномиальных алгоритмов имеет погрешность не более $(3 + \varepsilon)$, $\varepsilon > 0$, [53].

В перечисленных выше моделях размещения предполагается, что решение принимает одно лицо, стоящее на высшем уровне иерархии. Отчасти это объясняется тем, что многие задачи размещения возникли в период централизованного управления. В последнее время в условиях рыночной экономики более актуальными становятся ситуации, когда несколько лиц на разных уровнях иерархии участвуют в процессе принятия решений. В таких моделях участники процесса имеют собственные цели и предпочтения, которые могут быть противоположными, в результате чего возникают конфликтные ситуации. Процесс принятия решений в таких системах выглядит следующим образом. Первым принимает решение верхний уровень иерархии. Это решение передаётся нижестоящим и уже не меняется. Каждый уровень иерархии, получив решение вышестоящих уровней, принимает своё решение. Он преследует свои цели и использует имеющиеся у него возможности и ресурсы. Результат деятельности иерархической системы в целом зависит от работы всех уровней иерархии. Задача состоит в том, чтобы найти такое решение верхнего уровня, которое приводит систему к достижению глобальной цели.

Если верхний уровень полностью определяет поведение нижестоящих уровней, то моделирование процессов принятия решений упрощается. В этом случае возникают классические задачи математического программирования, в которых область допустимых значений задается набором равенств и неравенств. Если же верхний уровень может только влиять на работу нижних уровней, то возникает новый класс задач. Область допустимых значений в данном случае задается с помощью вспомогательных (внутренних) задач, моделирующих поведение нижних уровней иерархии. Решения вышестоящих органов выступают в этих задачах в качестве ограничений.

Задачи, в которых рассматриваются только два уровня иерархии, называются задачами двухуровневого программирования. Впервые такие задачи в игровой постановке рассматривались в [70]. В России над этими задачами работали Ю.Гермейер и его ученики [4]. Обзор современного состояния в области двухуровневого программирования можно найти, например, в [24, 73].

В диссертации рассматривается задача о p-медиане с предпочтениями клиентов (МПК). В этой задаче имеются два уровня принятия решений. Они неравноправны. Сначала на верхнем уровне принимается решение об открытии p предприятий. Затем на нижнем уровне, зная места расположения этих предприятий, клиенты самостоятельно выбирают поставщиков, руководствуясь собственными предпочтениями. Задача состоит в том, чтобы выбрать на верхнем уровне открываемые предприятия и обслужить всех клиентов с минимальными суммарными затратами, принимая во внимание тот факт, что поставщиков выбирают не на верхнем, а на нижнем уровне. Если предпочтения на нижнем уровне совпадают с предпочтениями на верхнем уровне, то получаем классическую задачу о p-медиане. Следовательно, задача МПК является NP-трудной в сильном смысле и не принадлежит классу APX [13].

Впервые задачи размещения с предпочтениями клиентов рассматривались П. Ханжоулем и Д. Петерсом [42]. Позже аналогичные модели независимо были построены В.Т. Дементьевым, Ю.В. Шамардиным и Л.Е. Горбачевской [6, 7, 8]. В [6] доказано, что если матрица транспортных затрат в целевой функции на верхнем уровне и матрица предпочтений клиентов на нижнем уровне обладают свойством связности, то такая задача решается эффективно. Если же матрица предпочтений является квазивыпуклой, то задача сводится к задаче о "ближайшем соседе" с фиксированным числом точек в оптимальном решении. В [8] исследовалась более общая задача без ограничения на число открываемых предприятий с условием единственности оптимального потребительского выбора. В [2] приводится сведéние общего случая задачи МПК без ограничения на число открываемых предприятий к задаче минимизации полиномов от булевых переменных. Это сведéние позволяет выявить полиномиально разрешимые случаи решения задачи. Вопрос построения полиномиальных приближенных алгоритмов с гарантированными оценками точности для метрической задачи МПК остаётся на сегодняшний день открытым.

Как обобщение классической задачи о p-медиане задача МПК относится к классу NP-трудных задач в сильном смысле. Поэтому для её решения часто используются различные эвристические алгоритмы. На сегодняшний день наиболее успешными при решении практических задач являются итерационные методы локального поиска. К ним можно отнести поиск с запретами [36, 37, 38, 64], поиск с чередующимися окрестностями [44], генетические алгоритмы [39], алгоритмы имитации отжига [20, 50] и другие метаэвристики [51, 33, 63, 65]. Многие из них используют стандартную процедуру локального спуска. Это процесс последовательного движения от текущего к соседнему решению с лучшим значением целевой функции. Процесс останавливается, когда достигнут локальный оптимум, то есть решение, которое не имеет соседнего решения с лучшим значением целевой функции. Предполагается, что для каждого решения задано множество соседних решений или окрестность. Задачу нахождения локального оптимума для заданной окрестности называют задачей локального поиска.

С теоретической точки зрения интересным представляется изучение вычислительной сложности алгоритмов локального поиска, анализ поведения алгоритма в среднем и худшем случаях. Эмпирические результаты показывают, что для многих NP-трудных задач [33, 36, 46, 63, 67] локальный поиск позволяет находить приближенные решения близкие по значению целевой функции к глобальному оптимуму. Причём трудоемкость в среднем часто оказывается полиномиальной. Однако для целого ряда окрестностей и при любом выборе направления спуска число шагов алгоритма в худшем случае, например, для решения задачи о p-медиане, не может быть ограничено сверху полиномом от длины записи исходных данных [13]. Для теоретического анализа вычислительной сложности задач локального поиска используется специальный класс PLS (сокращение от polynomial-time local search problems). С содержательной точки зрения этот класс содержит те задачи, в которых локальная оптимальность может быть проверена за полиномиальное время: для заданного решения требуется определить, является ли оно локальным оптимумом, и если нет, то найти соседнее решение с лучшим значением целевой функции. В этом случае соответствующую окрестность называют полиномиально проверяемой. Если предположить, что NP$\neq$co-NP, то в классе PLS нет NP-трудных задач [68, 75]. Другими словами, не существует NP-полных задач, которые за полиномиальное время сводились бы к какой-нибудь задаче локального поиска из класса PLS. Это утверждение показывает, что скорее всего класс PLS не содержит NP-трудных задач. Поэтому задачи локального поиска не так сложны, как NP-трудные, что, в частности, подтверждают и экспериментальные исследования. Нахождение локального оптимума для полиномиально проверяемой окрестности не представляется сложным делом на практике. В классе PLS, как и в классе NP, удалось выделить наиболее трудные задачи, PLS-полные задачи, для которых доказан аналог теоремы Кука [47]. Если существует полиномиальный алгоритм решения хотя бы одной PLS-полной задачи, то все задачи из класса PLS полиномиально разрешимы. Однако до сих пор неизвестно ни одного полиномиального алгоритма решения PLS-полных задач. Построение такого алгоритма было бы сенсацией. Все задачи из класса PLS оказались бы полиномиально разрешимыми. Как пишут Я.К. Ленстра и Т. Вредевельд [74] "..скорее всего это не так, потому что существование такого алгоритма потребовало бы разработки достаточно общего подхода для нахождения локальных оптимумов, не менее общего, чем метод эллипсоидов, так как задача линейного программирования с симплексной окрестностью принадлежит классу PLS".

Аналогия с линейным программированием здесь весьма уместна и может быть даже глубже, чем кажется. Рождение симплекс метода толкнуло вперёд разработку методов линейного программирования и сегодня это один из самых мощных методов, используемых в приложениях. Он не является полиномиальным при оценке поведения в худшем случае, но в среднем он работает очень эффективно [14, 16, 62]. Аналогичную картину можно наблюдать и с алгоритмами локального спуска. Установлено [75, 13], что в худшем случае для ряда PLS-полных задач алгоритм локального спуска требует экспоненциального числа итераций, но в среднем его поведение полиномиально. Это обстоятельство объясняет интерес к таким алгоритмам, исследование их возможностей и модификаций, а также применение в более сложных схемах метаэвристик. Именно эти вопросы являются центральными в диссертации и исследуются на примере задачи МПК.

Изложим кратко результаты диссертационной работы. Она состоит из введения, четырех глав, заключения и списка используемой литературы.

Как уже отмечалось, работа посвящена алгоритмам локального поиска для задачи МПК. В первой главе приводится математическая постановка задачи в виде задачи двухуровневого программирования и описываются ее свойства. Приводятся определение задачи локального поиска и схема стандартного алгоритма локального спуска. Существенное влияние на трудоемкость алгоритма и время его работы оказывает правило выбора направления спуска или правило замещения. В окрестности решения часто имеется несколько элементов для возможного направления спуска. Интуитивно кажется, что надо выбирать элемент с наилучшим значением целевой функции, но как показывают вычислительные эксперименты такой выбор не всегда является наилучшим. В связи с этим в работе исследовалось пять известных правил замещения и предложено новое правило замещения, которое приводит к локальным оптимумам с малой относительной погрешностью. Обсуждаются результаты экспериментальных исследований для сравнения правил замещения по точности и числу итераций при получении локальных оптимумов. Изучаются свойства ландшафта задачи и взаимное расположение локальных оптимумов. Напомним, что ландшафтом называют взвешенный ориентированный граф $G = (V, E)$, вершины которого соответствуют допустимым решениям задачи. Дуга (v_1, v_2), $v_1, v_2 \in V$ принадлежит E, если допустимое решение для v_2 является соседним решением для v_1. Веса в этом графе приписаны вершинам и равны соответствующим значениям целевой функции. Изучение ландшафтов позволяет предсказывать поведение методов локального поиска и, в частности, отвечать на следующие вопросы. Как быстро можно перейти от одного локального оптимума к другому? Как много шагов с ухудшением нужно сделать, чтобы алгоритмом локального спуска можно было получить другой (лучший) локальный оптимум? Правда ли, что из "плохого" локального оптимума легко найти путь к "хорошему"? Результаты экспериментов приведённые в первой главе, говорят о том, что ландшафт устроен таким образом, что стоит сделать один шаг с ухудшением даже из хорошего локального оптимума (погрешность около 5%) и можно получить лучший локальный оптимум, причём достаточно далеко от исходного. Использование больших окрестностей, типа окрестности Лина-Кернигана, также должно быть эффективно и улучшать качество решений для таких метаэвристик, как генетические алгоритмы, поиск с чередующимися окрестностями, алгоритмы муравьиных колоний и др. [33, 63].

Во второй главе проводится теоретический анализ вычислительной сложности алгоритмов локального поиска в худшем случае. Показана PLS-полнота задачи МПК с рядом полиномиально проверяемых окрестностей. Установлено, что нахождение локального минимума является PSPACE-полной задачей. Приводится сведе́ние задачи МПК к задаче минимизации полинома от булевых переменных. Впервые такое сведе́ние было предложено В.Л. Бересневым для простейшей задачи размещения [1]. Аналогичное сведе́ние для задачи МПК получено Л. Горбачевской [6]. Это сведе́ние используется для формулировки условий оптимальности Куна-Таккера, а также для связи локально-оптимальных решений с локальными седловыми точками функции Лагранжа. Установлено, что нахождение локальных седловых точек функции Лагранжа является столь же сложной задачей, что и нахождение локальных оптимумов.

В третьей главе для получения нижних оценок оптимума рассматриваются три известных сведе́ния задачи МПК к задачам целочисленного линейного программирования (ЦЛП) [6]. Предлагается новое сведе́ние, которое доминирует остальные по значению целевой функции линейной релаксации. Известно [8], что для задачи МПК можно построить сведе́ние к задаче с парой матриц с дополнительным ограничением на число выбираемых строк. Эта задача также может быть записана в терминах ЦЛП. Однако, такое представление не единственное. В данной работе исследуются два таких представления. Одно из них позволяет находить новую нижнюю оценку, которая не хуже предшествующих и может оказаться в произвольное число раз лучше их. Один из разделов третьей главы посвящен генетическим алгоритмам и исследованию влияния различных параметров алгоритма на качество получаемых решений. Приводятся результаты численных экспериментов на одном из сложных классов тестовых примеров из электронной библиотеки "Дискретные задачи размещения". Проводится сравнение построенных нижних оценок, нижней оценки получаемой решением классической задачи о p-медиане и верхних оценок, полученных разработанным генетическим алгоритмом. Величина разрыва между верхними и нижними оценками оказывается достаточно большой, что оставляет поле для дальнейших исследований и, в частности, для ответа на вопрос: верхние или нижние оценки приводят к появлению этого разрыва или он обусловлен природой самой задачи?

В четвёртой главе приводится описание системы поддержки принятия решений (СППР), разработанной в ходе выполнения в Институте математики им. С.Л. Соболева СО РАН прикладной НИР по заказу одной из крупных компаний, выпускающих машины для сельского хозяйства. Интеллектуальным ядром СППР является математическая модель двухуровневого целочисленного программирования, тесно связанная с задачей МПК. Использование разработанной СППР позволило сократить номенклатуру выпускаемых сельскохозяйственных машин и увеличить прибыль компании.

Полученные результаты опубликованы и докладывались на международном симпозиуме по исследованию операций "SYM-OP-IS-2003" (г. Герцег-Нови, Черногория, 2003), на международной конференции "Дискретный анализ и исследование операций"(г. Новосибирск, 2004), на международной конференции по исследованию операций (г. Тилбург, Голландия, 2004), на 13-м международной байкальской школе-семинаре (г. Северобайкальск, 2005), на II, III всероссийских конференциях "Проблемы оптимизации и экономические приложения"(г. Омск 2003, 2006), на 18-й мини-евро конференции по алгоритмам локального поиска с переменными окрестностями (г. Пуэрто-де-ля-Круз, Испания, 2005), на школа-семинаре "Проблемы оптимизации сложных систем"(г. Новосибирск, 2005, 2006), на научных семинарах в Институте математики им. С.Л. Соболева СО РАН.

Автор выражает глубокую признательность и благодарность научным руководителям Бересневу В.Л. и Кочетову Ю.А., Плясунову А.В., всем сотрудникам лаборатории "математические модели принятия решений" за помощь, постоянное внимание и всестороннюю поддержку. Фокину М.В. за постановку интересной прикладной задачи. Августиновичу С.В. за ценные идеи и замечания. А также коллегам из иркутского института динамики систем и теории управления СО РАН за предоставленные результаты расчетов.

Глава 1

Задача о p-медиане с предпочтениями клиентов

В классической задаче о p-медиане [54] заданы множество клиентов, множество предприятий и матрица производственно-транспортных затрат на обслуживание клиентов из этих предприятий. Требуется выбрать ровно p предприятий так, чтобы суммарные затраты на обслуживание клиентов были бы минимальными. Неявно в задаче предполагается, что клиенты согласны с любым выбором поставщиков и их мнением можно пренебречь. Содержательно более интересная ситуация возникает в случае, когда это не так, и мнения клиентов следует явным образом учитывать в математической модели.

В задаче о p-медиане с предпочтениями клиентов (МПК) имеются два уровня принятия решений. Они неравноправны. Сначала на верхнем уровне принимается решение об открытии p предприятий. Затем на нижнем уровне, зная места расположения этих предприятий, клиенты самостоятельно выбирают поставщиков, руководствуясь собственными предпочтениями. Задача состоит в том, чтобы выбрать на верхнем уровне открываемые предприятия и обслужить всех клиентов с минимальными суммарными затратами, принимая во внимание тот факт, что поставщиков выбирают не на верхнем, а на нижнем уровне. Если предпочтения на нижнем уровне совпадают с предпочтениями на верхнем уровне, то получаем классическую задачу о p-медиане [58, 59, 65]. Следовательно, задача МПК является NP-трудной в сильном смысле и не принадлежит классу APX [13].

1.1 Постановка задачи и ее свойства

Введём следующие обозначения:
$I = \{1, \dots, m\}$ – множество предприятий;
$J = \{1, \dots, n\}$ – множество клиентов;
$c_{ij} \geq 0, \ \ i \in I, j \in J,$ – транспортные затраты i-го предприятия при обслуживании j-го клиента;
$g_{ij} \geq 0,$ целые, $i \in I, j \in J,$ – предпочтения j-го клиента по отношению к i-му предприятию, если $g_{i_1 j} < g_{i_2 j}$, то j-й клиент из предприятий i_1, i_2 выберет предприятие i_1.
Переменные задачи $y = (y_i)$, $X = (x_{ij})$:
$y_i = \begin{cases} 1, & \text{если открывается } i\text{-е предприятие;} \\ 0 & \text{в противном случае;} \end{cases}$
$x_{ij} = \begin{cases} 1, & \text{если } j\text{-й клиент обслуживается } i\text{-м предприятием;} \\ 0 & \text{в противном случае.} \end{cases}$
С использованием введённых обозначений задача МПК может быть записана в виде следующей задачи двухуровневого программирования [8]: найти

$$\min_{X^*, y} \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij}(y) \tag{1.1}$$

$$\sum_{i \in I} y_i = p, \tag{1.2}$$

$$y_i \in \{0, 1\}, i \in I, \tag{1.3}$$

где $X^* = (x^*_{ij}(y))$ – оптимальное решение задачи клиентов:

$$\min_{X} \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} \tag{1.4}$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J, \tag{1.5}$$

$$x_{ij} \leq y_i, \quad i \in I, \quad j \in J, \tag{1.6}$$

$$x_{ij} \in \{0, 1\}, \quad i \in I, \quad j \in J. \tag{1.7}$$

Величина $\overline{F}(y, X^*) = \sum\limits_{i \in I} \sum\limits_{j \in J} c_{ij} x^*_{ij}(y)$ имеет смысл суммарных транспортных расходов, которые нужно минимизировать выбором набора из p предприятий. Этот набор описывается вектором $y = (y_i)$. Внутренняя экстремальная задача моделирует поведение клиентов, каждый из которых выбирает себе наиболее предпочтительное открытое предприятие. Этот выбор представляет матрица $X = (x_{ij})$. Для каждого значения переменной y верхнего уровня, ограничения на нижнем уровне определяют допустимое множество задачи клиентов

$$L(y) = \{X | \sum_{i \in I} x_{ij} = 1, \forall j \in J, x_{ij} \leq y_i, x_{ij} \in \{0, 1\}, i \in I, j \in J\}.$$

В работе предполагается, что выбор клиентов при каждом наборе y является однозначным. В этом случае множество оптимальных решений задачи нижнего уровня $L_{opt}(y) = argmin\{\sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} | X \in L(y)\}$ состоит из одного элемента. Чтобы гарантировать это условие достаточно предполагать, например, что $g_{ij} \neq g_{kj}$ для всех $i, k \in I$, $i \neq k$ и $j \in J$. Тогда функцию $\overline{F}(y, X^*)$ можно заменить на $F(y) = \min_{X \in L_{opt}(y)} \overline{F}(y, X)$ и задачу (1.1)–(1.7) можно записать в виде: найти

$$\min_y F(y) \tag{1.8}$$

$$\sum_{i \in I} y_i = p \tag{1.9}$$

$$y_i \in \{0, 1\}, i \in I. \tag{1.10}$$

В следующем параграфе будет показано, что некоторые задачи с неоднозначным выбором клиентов могут быть сведены к задаче с условием однозначности.

1.2 Задача МПК в условиях неоднозначности выбора клиентов

Более общей является ситуация, когда в задаче (1.4)–(1.7) может быть несколько оптимальных решений. В этом случае выбор клиентов становится неоднозначным. Введем вспомогательные обозначения: $I(y) = \{i \in I | y_i = 1\}$ – множество номеров открытых предприятий. Для каждого $j \in J$ определим $I_j(y) = \{i \in I(y) | g_{ij} = \min_{k \in I(y)} g_{kj}\}$ – множество открытых предприятий одинаковых, с точки зрения предпочтений j-го клиента. Пусть функция $H(I_j(y))$ определяет выбор j-ым клиентом предприятия из множества $I_j(y)$. Тогда в общем виде задача МПК может быть записана следующим образом: найти

$$F_H = \min_y \sum_{j \in J} c_{H(I_j(y))j} \tag{1.11}$$

$$\sum_{i \in I} y_i = p, \tag{1.12}$$

$$y_i \in \{0, 1\}, i \in I. \tag{1.13}$$

Иногда дополнительная информация о поведении клиентов известна и функция H определена однозначно. Например, среди оптимальных решений клиенты выбирают решение, которое доставляет наиме́ньшие суммарные транспортные затраты для поставщиков. Таким образом, клиенты сотрудничают с поставщиками, учитывая их интересы. Такую постановку задачи в литературе называют кооперативной [73, 34]. Математическая модель кооперативной постановки записывается следующим образом: найти

$$F_{\min} = \min_y \sum_{j \in J} \min_{i \in I_j(y)} c_{ij} \tag{1.14}$$

$$\sum_{i \in I} y_i = p, \tag{1.15}$$

$$y_i \in \{0, 1\}, i \in I. \tag{1.16}$$

Другим вариантом поведения клиентов может быть случай, когда среди оптимальных решений они выбирают решение, которое доставляет поставщикам наибо́льшие суммарные транспортные затраты. Такую постановку называют антикооперативной [73, 34]. Математическая модель антикооперативной постановки записывается следующим образом: найти

$$F_{\max} = \min_y \sum_{j \in J} \max_{i \in I_j(y)} c_{ij} \tag{1.17}$$

$$\sum_{i \in I} y_i = p, \tag{1.18}$$

$$y_i \in \{0, 1\}, i \in I. \tag{1.19}$$

Покажем, что кооперативная и антикооперативная задачи сводятся к задаче МПК с условием единственности оптимального выбора клиентов, т. е. к задаче (1.8)–(1.10).

Лемма 1 *Задача МПК в кооперативной постановке (1.14)–(1.16) сводится к задаче МПК с условием единственности выбора клиентов.*

Доказательство. Исходными данными кооперативной задачи (1.14)–(1.16) являются: множества I, J, матрицы (c_{ij}), (g_{ij}) и число p. Построим по этим данным исходные данные задачи (1.8)–(1.10). Точнее, по матрице (g_{ij}) построим новую матрицу (g'_{ij}), в каждом столбце которой будут разные элементы, остальные исходные данные останутся без изменения. Для каждого столбца $j \in J$ матрицы (g_{ij}) на всех элементах множества I определим перестановку $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ такую, что $g_{\pi_1 j} \leq g_{\pi_2 j} \leq \cdots \leq g_{\pi_m j}$, а для равных элементов $g_{\pi_i j} = g_{\pi_{i+1} j}$, определим перестановку π так, чтобы $c_{\pi_i j} \leq c_{\pi_{i+1} j}$. Элементы новой матрицы (g'_{ij}) зададим следующим образом: $g'_{\pi_i j} = i$, для всех $i \in I, j \in J$. По построению элементы каждого столбца матрицы (g'_{ij}) различны. Им присвоены значения, согласованные с матрицей транспортных затрат (c_{ij}). Таким образом, при каждом наборе y множество $L_{opt}(y)$ состоит из одного элемента, и матрица выбора X будет однозначно определяться по y. Оптимальное решение y^* в задаче (1.8)–(1.10) с исходными данными определенными выше будет оптимальным решением и для задачи (1.14)–(1.16) по построению матрицы (g'_{ij}). Лемма доказана.

Аналогичное сведение можно построить для антикооперативной постановки задачи.

Лемма 2 *Задача МПК в антикооперативной постановке (1.17)–(1.19) сводится к задаче МПК с условием единственности выбора клиентов.*

Доказательство леммы аналогично предыдущему с заменой в неравенстве $c_{\pi_i j} \leq c_{\pi_{i+1} j}$ знака " $\leq$ " на " $\geq$ ".

Если же о поведении клиентов заранее ничего не известно, то в этом случае можно указать интервал, которому принадлежит оптимальное значение целевой функции такой задачи.

Лемма 3 $F_{\min} \leq F_H \leq F_{\max}$.

Доказательство. Заранее неизвестно какое значение из множества $I_j(y)$ будет выбрано функцией H при заданном векторе y, но для всех $j \in J, i \in I_j$ справедливы неравенства:

$$\min_{i \in I_j(y)} c_{ij} \leq c_{H(I_j(y))j} \leq \max_{i \in I_j(y)} c_{ij}.$$

Следовательно,

$$\sum_{j\in J}\min_{i\in I_j(y)} c_{ij} \le \sum_{j\in J} c_{H(I_j(y))j} \le \sum_{j\in J}\max_{i\in I_j(y)} c_{ij},$$

т. е. $F_{\min} \le F_H \le F_{\max}$. Лемма доказана.

Учитывая приведенные выше сведéния и оценки, в остальной части работы будет рассматриваться задача МПК только в постановке с условием единственности выбора клиентов.

1.3 Алгоритм локального спуска и правила замещения

Исследование полиномиальных алгоритмов для задачи МПК интересно только для специальных классов исходных данных. В общем случае для задачи МПК целесообразно использовать неполиномиальные алгоритмы и, в частности, итерационные алгоритмы локального поиска.

Определение 1 *[26] В общем виде оптимизационная задача OP определяется следующим набором объектов $<\mathcal{I}, Sol, f, goal>$, где*

$\mathcal{I}$ – множество входов задачи OP;

Sol – функция, которая каждому входу $z \in \mathcal{I}$ сопоставляет множество допустимых решений $Sol(z)$;

f – функция, которая каждому допустимому решению s на входе z задаёт вес $f(s,z)$;

величина $goal \in \{min, max\}$ уточняет: является ли задача OP задачей на максимум или минимум.

В задаче OP необходимо найти оптимальное решение для заданного входа П.

В соответствии с этим определением входом z в задаче МПК являются матрицы $(c_{ij}), (g_{ij}), i \in I, j \in J$ и число p. Множество допустимых решений $Sol(z)$ состоит из пар $(y, X^*(y))$, удовлетворяющих системе ограничений (1.2)–(1.7). В качестве функции f выступает целевая функция задачи верхнего уровня.

Определение 2 *[26] Задача локального поиска – это пара $L = (OP, N)$, где OP – оптимизационная задача, а N – функция, которая каждому* *допустимому решению s на входе z ставит в соответствие множество $N(s,z) \subseteq Sol(z)$ соседних решений. Множество $N(s,z)$ называют окрестностью решения s. Задача локального поиска заключается в отыскании локального минимума для заданного входа z.*

Пусть z – вход задачи МПК, $(y, X^*(y)) \in Sol(z)$. В работе рассматриваются следующие окрестности.

1. Окрестность $N_1(y,z) = \{y' \in Sol(z))|d(y,y') = 2\}$, где d– расстояние Хэмминга. Окрестность содержит $p(m-p)$ допустимых решений, которые получаются закрытием одного предприятия и открытием другого.

2. Окрестность $N_{LK}(y,z)$ (сокращение от Lin-Kernighan) [49, 51] строится с помощью следующей итерационной процедуры. На каждом шаге просматриваются все пары элементов (i_1, i_2) такие, что $y_{i_1} = 0, y_{i_2} = 1$ и каждый элемент не использовался на предыдущих шагах. Затем, среди просмотренных пар выбирается наилучшая, т. е. пара, которая максимально уменьшает целевую функцию задачи, либо, если таких нет, минимальным образом увеличивает её. Если таких пар несколько, то выбирается любая из них. Процедура останавливается через $\min\{p, m-p\}$ шагов и дает $\min\{p, m-p\}$ соседних решений.

3. Окрестность $N_{LK_1}(y,z)$ состоит из одного подмножества, которое получается за один шаг предыдущей процедуры.

4. Окрестность $N_{FM}(y,z)$ (сокращение от Fiduccia-Mattheyses) [35]. Строится аналогично окрестности $N_{LK}(y,z)$. Каждый шаг этой процедуры состоит из двух этапов. На первом этапе просматриваются элементы множества $\{i \in I|y_i = 1\}$, каждый из которых ранее не использовался, и выбирается наилучший i_1. Затем полагается $y_{i_1} = 0$. На втором этапе среди элементов $\{i \in I|y_i = 0\}$, каждый из которых ранее не использовался, выбирается наилучший элемент i_2 и y_{i_2} полагается равным единице. Наилучшим элементом i по-прежнему считается тот, для которого изменение значения y_i на противоположное максимально уменьшает значение целевой функции, либо, если таких нет, минимальным образом увеличивает его. Если наилучших элементов несколько, то выбирается любой из них. Окрестность содержит $\min\{p, |I|-p\}$ допустимых решений.

5. Окрестность $N_{FM_1}(y,z)$. Соседнее по этой окрестности решение строится за один шаг предшествующей процедуры. Данная окрестность была

введена для задачи $Max - Graph\,Partitioning$ в [68] под названием FM-$Swap$.

6. Окрестность $N_k(y,z)$ [74]. Выбирается не более k пар (i_1, i_2), таких что $y_{i_1} = 1, y_{i_2} = 0$, не содержащих общих элементов, и для каждой из них производится замена. Мощность такой окрестности имеет порядок $O(p^k(|I| - p)^k)$, где k – константа.

Все перечисленные окрестности являются полиномиально проверяемыми.

Так как рассматриваемая в работе задача на минимум, то все определения приводятся для задач минимизации, хотя они могут быть перенесены и для задач на максимум.

Определение 3 *Допустимое решение (y^*, X^*) для входа z является локальным минимумом в задаче (1.1)–(1.7), если выполнены следующие условия*

1. *Для любого $X \in L_{opt}(y^*)$ $\overline{F}(y^*, X^*) \leq \overline{F}(y^*, X)$;*
2. *Для любой пары (y, X), где $(y, X) \in N((y^*, X^*), z), X \in L_{opt}(y)$ $\overline{F}(y^*, X^*) \leq \overline{F}(y, X)$.*

Благодаря предположению об однозначности определения решения нижнего уровня, при котором задача (1.1)–(1.7) сводится к задаче (1.8)–(1.10) можно использовать стандартное определение локального минимума.

Определение 4 *Допустимое решение y^* для входа z является локальным минимумом в задаче (1.8)–(1.10), если $F(y^*) \leq F(y)$ для любого $y \in N(y^*, z)$*

Выделим подмножество $N^*(y,z) \subseteq N(y,z)$ соседних решений с меньшим значением целевой функции, то есть $N^*(y,z) = \{y' \in N(y,z) \mid F(y') < F(y)\}$.

Стандартный алгоритм локального спуска можно представить следующей последовательностью шагов.

Шаг 0. Найти решение $y^0 \in Sol(z)$.
Шаг i. Если $N^*(y^{i-1}, z) \neq \emptyset$, то
    а) выбрать $y' \in N^*(y^{i-1}, z)$,
    б) положить $y^i := y'$,

в) перейти на шаг $(i+1)$,

Иначе завершить работу алгоритма.

При уточнении способа выбора соседнего решения y' возникает конкретный алгоритм локального спуска. Как показывают вычислительные эксперименты этот выбор оказывает существенное влияние на погрешность получаемых локальных оптимумов, а также число шагов алгоритма и время счета. Поэтому при реализации локального спуска этому пункту стоит уделить особое внимание. Существует несколько известных правил замещения.

1) *Спуск в направлении наилучшего элемента.* На каждом шаге локального спуска в множестве N^* выбирается допустимое решение с наименьшим значением целевой функции. Такой выбор кажется наиболее естественным, но требует просмотра всей окрестности.

2) *Спуск в направлении наихудшего элемента.* В некотором смысле это правило противоположно предыдущему, так как в множестве N^* выбирается элемент с наибольшим значением целевой функции. Такая стратегия дает наиболее пологий спуск к локальному минимуму, но может потребовать значительно больше шагов, чем в предыдущем случае.

3) *Спуск в направлении случайного элемента* предполагает выбор элемента из множества N^* случайным образом, например, с равномерным распределением.

4) *Спуск в направлении первый подходящий.* Поиск соседнего решения завершается, как только обнаружен первый элемент из множества N^*. В отличие от предыдущих правил в данном случае не требуется просмотра всей окрестности. По смыслу это правило близко к правилу *случайного элемента*, если просмотр окрестности начинается со случайной точки. Однако, чаще всего используется один и тот же порядок, например, лексикографический, и просмотр начинают с наименьшего элемента.

5) *Спуск по круговому правилу* [14]. Просмотр окрестности начинается с того места, где был найден элемент y на предшествующем шаге, а заканчивается на первом найденном элементе из множества N^*. Это правило основано на том наблюдении, что для многих задач при переходе к соседнему решению значение целевой функции в окрестности меняется незна-

чительно. Скорее всего, невыгодные решения в окрестности $N(y)$ будут невыгодными и в окрестности соседнего решения. Поэтому лучше продолжить просмотр, чем повторять его сначала. Конечно, данные соображения нужно проверять для каждой задачи отдельно.

В данной работе предлагается новое правило, основанное на идее выбора соседнего решения с наибольшим числом возможных направлений для спуска на следующем шаге алгоритма.

6) *Спуск по правилу максимальной свободы.* Для каждого элемента $y' \in N^*(y)$ вычисляется мощность множества $N^*(y')$, так называемая "свобода", а затем в множестве $N^*(y)$ выбирается элемент с наибольшей свободой для дальнейшего движения вниз.

1.4 Экспериментальные исследования

1.4.1 Влияние правил замещения на качество локальных оптимумов и число итераций алгоритма

Для сравнения правил был проведён вычислительный эксперимент на случайных матрицах. Цель эксперимента состояла в сравнении правил замещения по точности и числу итераций для получения локальных оптимумов, а также в выработке рекомендаций по применению того или иного правила в разрабатываемых алгоритмах. Каждый элемент матрицы (c_{ij}) выбирался из интервала $(0, \ldots, 1000)$ с равномерным распределением и независимо от других элементов. Матрица приоритетов (g_{ij}) на 70% совпадала с матрицей (c_{ij}) и в каждом столбце содержала только различные элементы. Для каждого $m = 30, 40, ..., 200$ генерировалось 10 тестовых примеров с $p = m/10, n = m$. Для каждого из них порождалось 1000 случайных допустимых решений. К каждому допустимому решению применялась стандартная процедура локального спуска с окрестностью N_1 и одним из шести правил замещения. Таким образом, для каждого примера получалось 6000 локальных оптимумов. Лучшее среди них обозначим через y^*. Поскольку точное решение задачи даже при небольшой размерности, $n = m = 50$, не удается найти с помощью коммерческого программного обеспечения, в основе решения которого лежит метод ветвей и границ, то узнать отклонение y^* от оптимума не представляется возможным. Однако, для сравнения

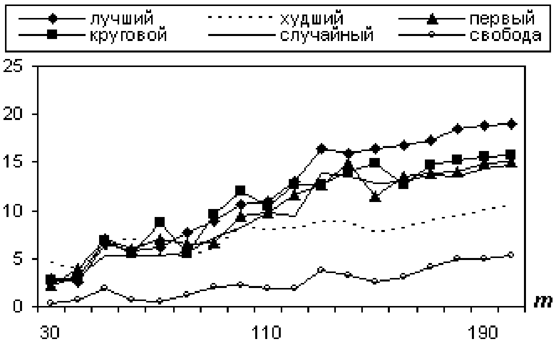

Рис. 1.1: Средняя относительная погрешность (%), $p = m/10$

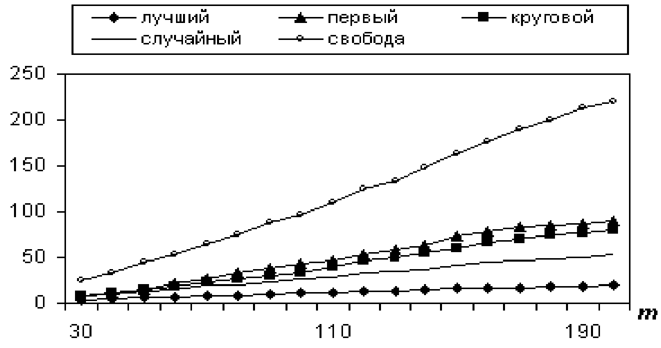

Рис. 1.2: Среднее число шагов, $p = m/10$

правил замещения этого не требуется. На рис. 1.1 показана средняя погрешность $\varepsilon = (F(y) - F(y^*))/F(y^*)$ получаемых локальных оптимумов относительно y^*. Каждая точка на графике показывает среднее значение по 10^4 испытаниям. Первое правило приводит к наибольшей погрешности. Правила 3–5 дают примерно одинаковые результаты. Наименьшая погрешность соответствует последнему правилу, которое является более трудоемким, чем предыдущие.

Зависимость среднего числа шагов локального спуска от размерности задачи показана на рис. 1.2, 1.3. Для всех правил, кроме второго, число шагов растет как линейная функция. Второе правило приводит к наибольшему числу шагов локального спуска. Так, например, при $n = m = 200$, среднее число шагов для шестого правила не превышает 250, в то время, как для второго правила оно не меньше 8000. Таким образом, выбор правила замещения играет важную роль как с точки зрения трудоемкости, так и точности получаемых локальных оптимумов. По-видимому, правила 3–4 являются вполне подходящими для применения, например, в генетических алгоритмах.

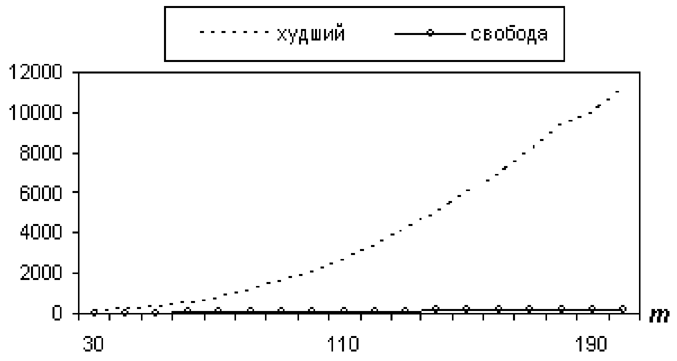

Рис. 1.3: Среднее число шагов, $p = m/10$

1.4.2 Исследование расположения локальных оптимумов

Чтобы исследовать возможности локального поиска для задачи МПК был проведен следующий эксперимент. Его цель состояла в исследовании свойств ландшафта задачи и взаимного расположения локальных оптимумов. Напомним, что под ландшафтом [75] понимают взвешенный ориентированный граф, вершины которого соответствуют допустимым решениям, а дуга $e = (v_1, v_2)$ присутствует в графе тогда и только тогда, когда v_2 является соседом v_1 и $F(v_1) > F(v_2)$ (для задач на максимум используется обратное неравенство). Локальные оптимумы оптимизационной задачи соответствуют стокам в этом графе. Если генетический алгоритм "работает" только с локальными оптимумами, то возникают следующие вопросы. Как быстро можно перейти от одного локального оптимума к другому? Как много шагов с ухудшением нужно сделать чтобы алгоритмом локального спуска можно было получить другой (лучший) локальный оптимум? Правда ли, что из "плохого" локального оптимума легко найти путь к "хорошему"?

Для ответа на эти вопросы использовалась следующая процедура. Пусть y – локальный оптимум для окрестности N_1 и y'– соседнее решение, отличающееся по координатам i_1 и i_2. Применим к решению y' процедуру локального спуска, не меняя этих координат. К полученному таким образом условному локальному оптимуму снова применим процедуру локального спуска, убрав запрет на i_1 и i_2. В результате получим локальный оптимум y'', который может совпадать с y, может быть лучше или хуже его. Применим эту процедуру к каждому соседнему решению y' и для определенности используем четвертое правило замещения. Диаграммы на рис. 1.4 (1.4а–1.4е) показывают среднее число различных (всех) локальных оптимумов с

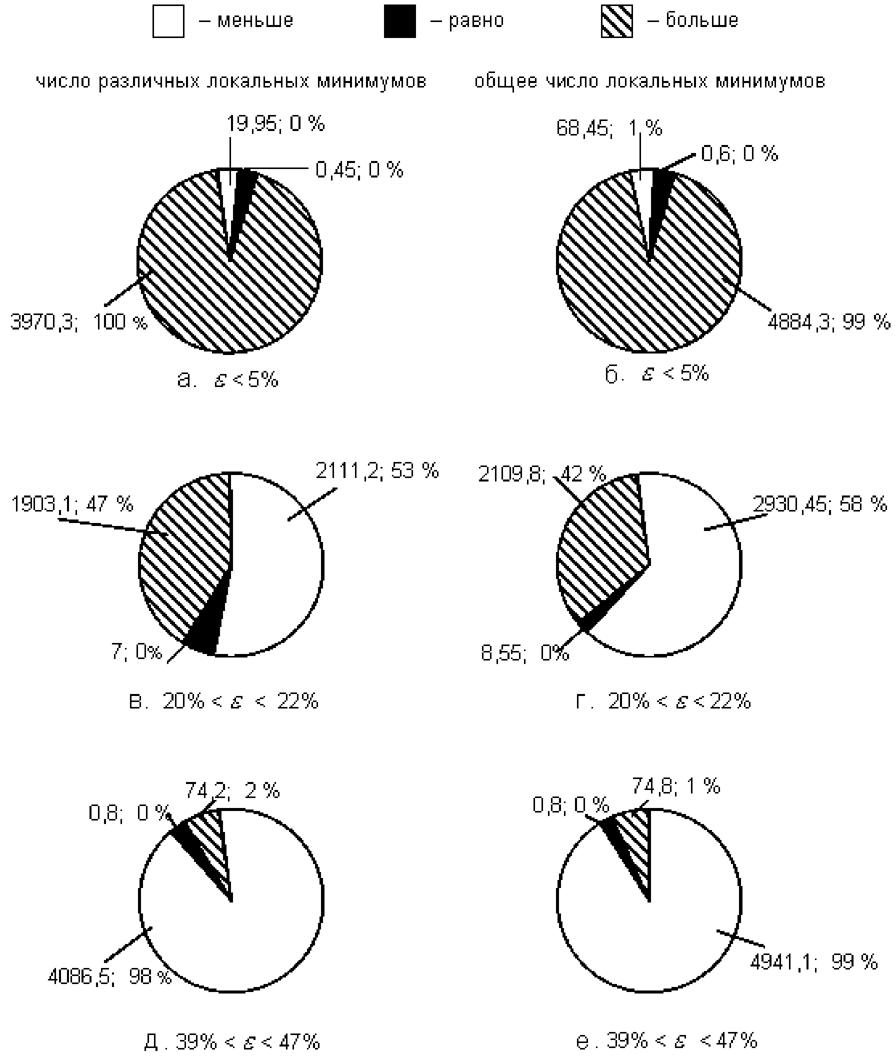

Рис. 1.4: Расположение локальных минимумов, $m = n = 250, p = 25$

меньшим, равным и большим значением целевой функции в абсолютных числах и процентах для одного из наиболее трудных примеров библиотеки Резенде, Вернека [65] при $n = m = 250, p = 25$. Матрица (g_{ij}) строилась так же, как и в предыдущем случае.

Диаграммы 1.4а, 1.4б соответствуют малой погрешности $\varepsilon < 5\%$ стартового локального оптимума. Диаграммы 1.4в, 1.4г и 1.4д, 1.4е — средней $20\% < \varepsilon < 22\%$ и большой $39\% < \varepsilon < 47\%$ погрешностям. Результаты проведенного эксперимента показывают, что чем больше погрешность локального оптимума, тем легче найти с помощью данной процедуры лучший локальный оптимум. Даже при малой погрешности это удается сделать, хотя процент лучших локальных оптимумов оказывается малым (см. рис. 1.4а, 1.4б). При большой погрешности (см. рис. 1.4д, 1.4е) почти каждый но-

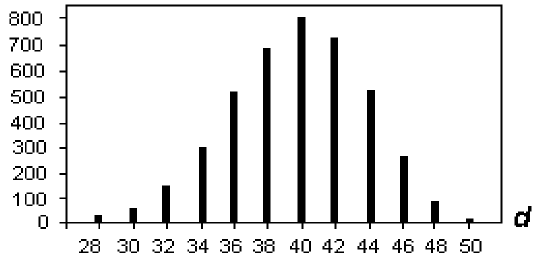

Рис. 1.5: Среднее число локальных минимумов на расстоянии d, $\varepsilon < 5\%$

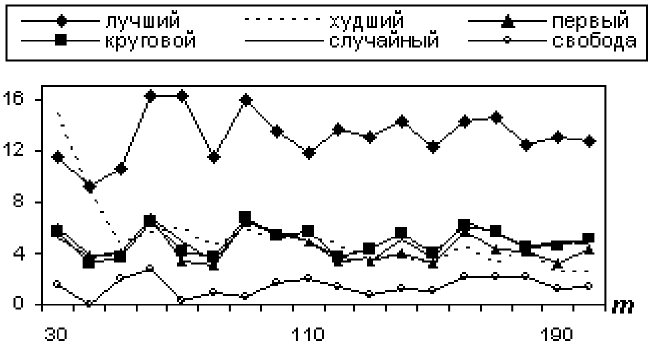

Рис. 1.6: Средняя относительная погрешность(%), $p = m/10$

вый локальный оптимум имеет меньшую погрешность, чем y. Рисунок 1.5 показывает среднее число получаемых локальных оптимумов на заданном расстоянии от y. Интересно отметить, что наибольшее число локальных оптимумов оказалось на расстоянии 40 при диаметре допустимой области 50 и не нашлось ни одного локального оптимума ближе 28. Другими словами, ландшафт в данном примере устроен таким образом, что стоит сделать один шаг с ухудшением даже из хорошего локального оптимума ($\varepsilon < 5\%$) и можно получить новый локальный оптимум достаточно далеко от исходного. Остановка в произвольном локальном оптимуме может приводить к большим погрешностям, но использование больших окрестностей, типа окрестности N_{LK} должно легко исправлять данный недостаток.

1.4.3 Сравнительный анализ с классической задачей о p-медиане

Вышеописанные эксперименты проводились и для классической задачи о p-медиане. Исходных данных были взяты из пункта 1.4.1, не учитывая предпочтения клиентов. Как и раньше относительная погрешность по первому правилу наибольшая (см. рис. 1.6), причем, разрыв в погрешности между первым правилом и остальными стал более заметным и составляет около

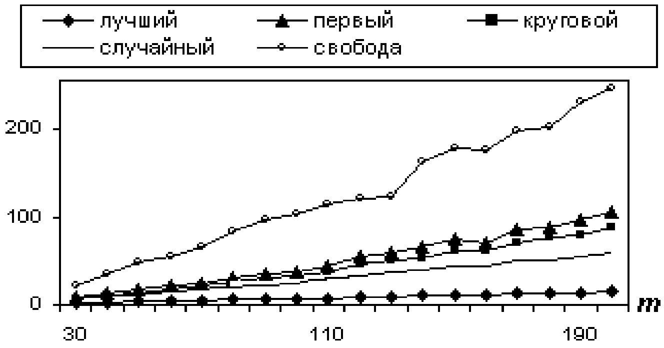

Рис. 1.7: Среднее число шагов, $p = m/10$

8%. По-видимому, этот эффект объясняется влиянием предпочтений клиентов в задаче МПК, из-за чего нахождение в той задаче "хороших" локальных минимумов по правилам 1)–5) усложняется. Правило максимальной свободы по-прежнему приводит к решениям с наименьшей погрешностью. Скорость роста среднего числа шагов осталась линейной для правил 1, 3–5 и квадратичной для правила 2 (см. рис. 1.7, 1.8).

Для сравнения ландшафтов проводился эксперимент из пункта 1.4.2 на том же примере из библиотеки Резенде, Вернека с $g_{ij} = c_{ij}$, $i \in I, j \in J$. Диаграммы (1.9а)–(1.9г) показывают, что сделав шаг с ухудшением из локального оптимума с небольшой погрешностью ($\varepsilon \leq 1\%$) можно найти еще 9% новых локальных оптимумов с меньшим значением целевой функции, если же погрешность стартовой точки меньше 4%, то число новых, более хороших локальных минимумов возрастает до 13%. Количество новых различных найденных решений составило от 9% до 33% от мощности всей окрестности. Этот показатель существенно меньше, чем для задачи МПК, что еще раз подтверждает влияние внутренней подзадачи клиента. Диаграмма 1.10 показывает, что наибольшее число локальных минимумов расположено близко ($d \leq 28$), причем их большая часть сконцентрирована на расстоянии шести парных замен ($d = 12$).

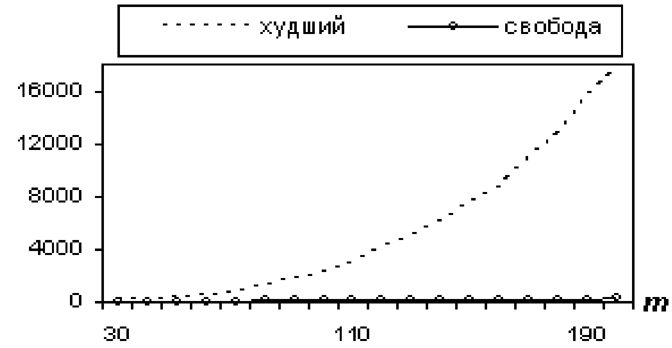

Рис. 1.8: Среднее число шагов, $p = m/10$

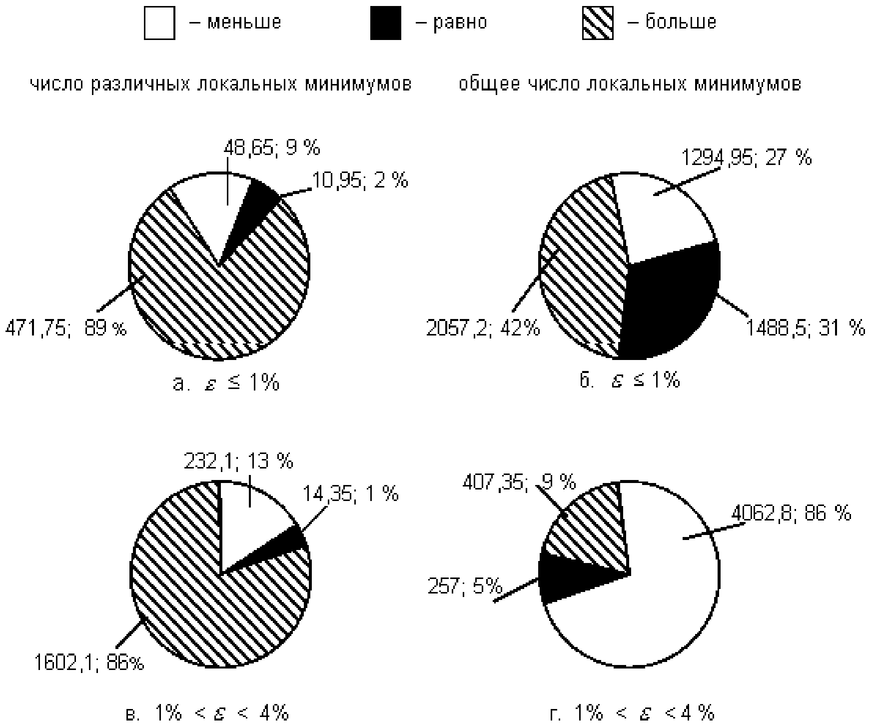

Рис. 1.9: Расположение локальных минимумов, $m = n = 250, p = 25$

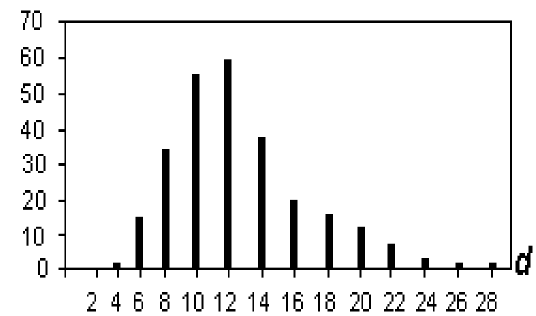

Рис. 1.10: Среднее число локальных минимумов на расстоянии d, $\varepsilon \leq 1\%$

Глава 2

Локальные оптимумы и их свойства

В данной главе проводится теоретический анализ вычислительной сложности алгоритма локального поиска в худшем случае. В разделе 2.1 приводится описание класса PLS, сводимости задач в этом классе, PLS-полнота задачи МПК с рядом полиномиально проверяемых окрестностей. В разделах 2.2 и 2.3 обсуждается вычислительная сложность алгоритмов локального спуска в худшем и среднем случаях. Раздел 2.4 посвящен обзору теоретических результатов о погрешности получаемых локальных оптимумов. Далее, в разделе 2.5 приводится сведение задачи МПК к задаче минимизации полинома от булевых переменных, доказываются условия оптимальности Куна-Таккера, показана связь локальнооптимальных решений с локально-седловыми точками функции Лагранжа.

2.1 Сложность задач локального поиска

Как уже отмечалось во введении, задачи локального поиска представляют нетривиальный класс задач комбинаторной оптимизации. Приведем точные определения и сформулируем основные утверждения для задач локального поиска применительно к задаче МПК.

Пусть $\Pi = (OP, N)$ — задача локального поиска, такая что длина любого допустимого решения задачи OP ограничена сверху некоторым полиномом от длины записи исходных данных.

Если s –локальный минимум в задаче Π, т. е. в его окрестности нет решений с меньшим значением целевой функции, то говорят, что решение s является N-*оптимальным.*

Определение 5 *[75] Задача* Π *принадлежит классу PLS, если существует три полиномиальных алгоритма A, B и C со следующими свойствами:*

1. Алгоритм A определяет, является ли слово z *входом задачи. Если* $z \in \mathcal{I}$*, то алгоритм находит допустимое решение задачи* OP.

2. Алгоритм B для любого входа задачи $z \in \mathcal{I}$ *и любого слова* s *определяет, является ли* s *допустимым решением. Если* $s \in Sol(z)$*, то алгоритм за полиномиальное время находит значение целевой функции.*

3. Алгоритм C для любого входа $z \in \mathcal{I}$ *и любого решения* $s \in Sol(z)$ *определяет, является ли* s *локальным оптимумом. Если нет, то алгоритм находит соседа* $s' \in N(s, z)$ *с меньшим значением целевой функции.*

Класс PLS не является пустым. Многие оптимизационные задачи с полиномиально проверяемыми окрестностями принадлежат этому классу. В частности, задача МПК с окрестностью N_1 также принадлежит классу PLS. Выполнение первых двух свойств из определения очевидно, поскольку допустимым решением задачи является любой набор из p предприятий, а оптимальное решение внутренней подзадачи восстанавливается по этому набору однозначно за полиномиальное время. Существование алгоритма C тоже очевидно, поскольку окрестность N_1 содержит $p(m - p)$ элементов. Значит просмотр всей окрестности и нахождение наилучшего решения можно выполнить за полиномиальное время. Обобщенная задача о назначениях [57] и полностью целочисленная задача размещения с ограничениями на мощности производства [29] не принадлежат этому классу, т. к. для этих задач поиск допустимого решения является NP-полной задачей.

Для задач из класса PLS легко построить алгоритмы локального улучшения. 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 оптимумом. Заметим, что алгоритм C допускает любое правило замещения. Меняя правила, получаем различные алгоритмы локального поиска. Для некоторых задач такие алгоритмы являются полиномиальными.

В классе PLS содержатся полиномиально разрешимые задачи, т. е. задачи, в которых за полиномиальное время можно найти локальный оптимум. Многие комбинаторные задачи без весовых функций с любой полиномиально проверяемой окрестностью относятся к таковым: задачи о клике, о покрытии, о независимом множестве и другие. Однако, в общем случае вопрос о вычислительной сложности нахождения локального оптимума остается открытым. Если предположить, что $NP \neq co - NP$, то в классе PLS нет NP-трудных задач.

Теорема 1 *[68] 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 [62]. 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 [9].

Для задач локального поиска определяется понятие PLS-сведения.

Определение 6 *[75] Пусть Π_1 и Π_2 — две задачи локального поиска. PLS-сведение задачи Π_1 к задаче Π_2 состоит в построении двух полиномиально вычислимых функций h и g таких, что*

1. по произвольному входу z задачи Π_1 функция h вычисляет некоторый вход $h(z)$ задачи Π_2;

2. по произвольному решению s для входа $h(z)$ функция g находит некоторое решение $g(s, z)$ для входа z;

3. для всех $z \in \Pi_1$, если s — локальный оптимум для входа $h(z) \in \Pi_2$, то $g(s, z)$ — локальный оптимум для входа z.

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 7 *[75] Задачу Π из класса PLS называют PLS-полной, если любая задача из класса PLS может быть PLS-сведена к ней.*

Полные задачи являются наиболее трудными в данном классе, и если хотя бы одна из них может быть решена за полиномиальное время, то и все остальные задачи могут быть решены за полиномиальное время.

Первая PLS-полная задача $Circuit$, как и в случае класса NP, была обнаружена в схемах из функциональных элементов [47]. Эта задача локального поиска формулируется для схемы z с m входами $(x_1, ..., x_m)$ и n выходами $(y_1, ..., y_n)$. Схема состоит из полиномиального числа элементов AND, OR, NOT. Множество ее допустимых решений $Sol(z)$ состоит из всех булевых векторов длины m. Целевая функция $f(x)$ определяется следующим образом:

$$f(x) = \sum_{j=1}^{n} 2^{j-1} y_j,$$

где y_j является j-м входом схемы z. Окрестность $N_{Flip}(x)$ решения x состоит из всех булевых векторов длины m, имеющих расстояние Хэмминга равным 1 от x. Максимизационная и минимизационная версии этой задачи являются PLS-полными [75]. PLS-полнота установлена для задач: о максимальном разрезе [68], о выполнимости на максимум с окрестностью N_{Flip} [56], о разбиении графа на две равные части с окрестностями N_1, N_{LK}, их модификациями и другими [68, 47], задача коммивояжера с окрестностью N_{2-opt} [55]. На рис.2.1 даны последовательности PLS-сведений для известных PLS-полных задач.

По аналогии с классом P полиномиально разрешимых оптимизационных задач введен класс P_{PLS} полиномиально разрешимых задач локального поиска.

Определение 8 *[26] Задача* $\Pi = (OP, N) \in PLS$ *принадлежит классу* P_{PLS}, *если существует полиномиальный алгоритм, который по любому входу* z *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. Геометрически, для вершины многогранника соседними вершинами являются те вершины, которые соединены ребром с данной вершиной. Алгебраически, если предполагать невырожденность задачи, для данного базиса соседними будут все допустимые базисы, получающиеся

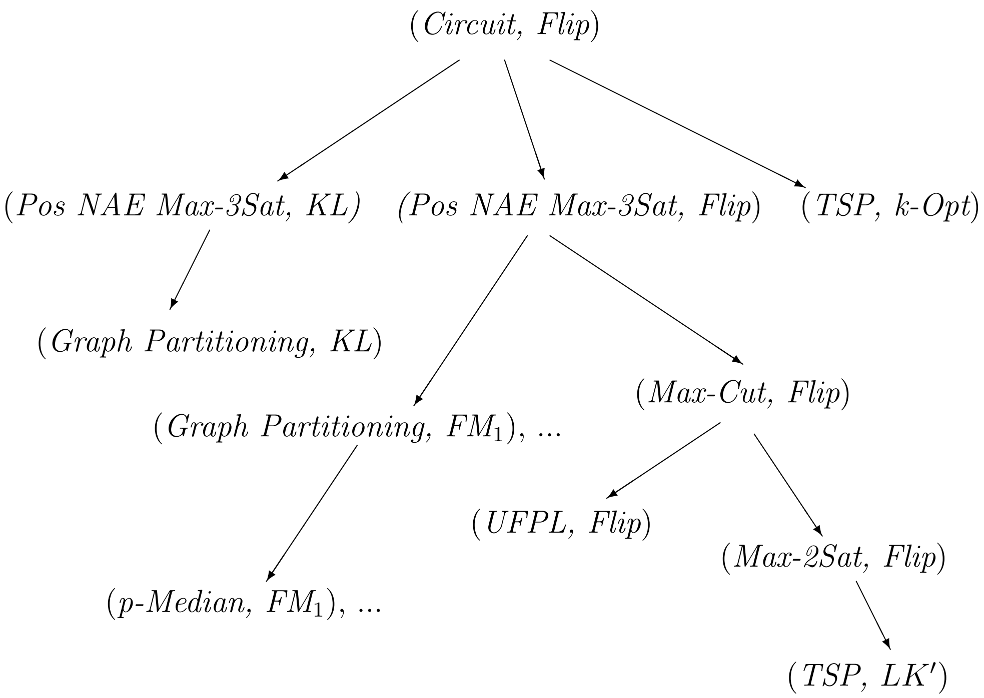

Рис. 2.1: PLS-полные задачи

из данного заменой одной базисной переменной на небазисную. Несмотря на то, что симплекс-метод не является полиномиальным алгоритмом существование метода эллипсоидов, который позволяет за полиномиальное время находить точное решение задачи, влечет принадлежность задачи к классу P_{PLS}.

В [75] между классом PLS и классами P_S и NP_S была установлена следующая цепочка вложений

$$P_S \subseteq PLS \subseteq NP_S,$$

где P_S и NP_S – формальные аналоги класса задач поиска решаемых за полиномиальное время на детерминированных и недетерминированных машинах Тьюринга соответственно. Первое включение означает, что любая полиномиальноразрешимая 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 [75], что данное включение является строгим, если $NP \neq co - NP$. 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. В [75] показано, что класс P_{PLS} эквивалентен классу P_S. 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.

2.2 Временная сложность алгоритмов локального поиска

Как отмечалось выше, принадлежность задачи локального поиска к классу PLS гарантирует возможность применения стандартного алгоритма локального спуска, каждый шаг которого является полиномиальным. Ниже будет показано, что в худшем случае этому алгоритму требуется экспоненциальное число шагов для достижения локального оптимума в задаче МПК с окрестностями N_{FM_1}.

Определение 9 *[75] Графом переходов $TG_\Pi(z)$ для входа z задачи* Π *называется ориентированный граф с множеством вершин $Sol(z)$ и множеством дуг вида (s, s'), где $s' \in N(s, z)$ и $f(s', z) < f(s, z)$. Высота вершины s есть длина кратчайшего пути в графе $TG_\Pi(z)$ из вершины s в сток (т.е. в локальный минимум). Высота графа $TG_\Pi(z)$ равна максимальной высоте его вершин.*

Из определения следует, что граф переходов является ациклическим и может иметь экспоненциальное число вершин относительно длины входа z. Если s не является локальным оптимумом, то переходу от s к s' в графе $TG_\Pi(z)$ соответствует ребро (s, s'). Применяя стандартный 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 до 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 10 *[75] Пусть* Π_1 *и* Π_2 *– две задачи локального поиска и* (h, g) *— PLS-сведение* Π_1 *к* Π_2*. Такое сведение называется плотным, если для любого входа* z *задачи* Π_1 *высота графа* $TG_{\Pi_2}(h(z))$ *не меньше высоты графа* $TG_{\Pi_1}(z)$*.*

Плотные сведéния позволяют переносить нижние оценки числа шагов алгоритма локального спуска с одной задачи на другую. Если алгоритму требуется экспоненциальное число шагов для решения задачи Π_1 и задача Π_1 плотно сводится к задаче Π_2, то и для задачи Π_2 алгоритму также потребуется экспоненциальное число шагов. В [75] доказаны достаточные условия, при которых PLS-сведéние будет плотным.

Теорема 2 *[75] Пусть* Π_1 *и* Π_2 *— две задачи локального поиска и* (h, g) *— PLS-сведéние* Π_1 *к* Π_2*. Данное сведéние является плотным, если для любого входа* z *задачи* Π_1 *существует подмножество* R *допустимых решений задачи* Π_2 *на входе* $y = h(z)$ *такое, что*
1. в R *содержатся все локальные минимумы для входа* y*;*
2. существует полиномиальный алгоритм, который для каждого решения t *на входе* z *позволяет находить такое решение* $q \in R$ *для входа* y*, что* $g(q, z) = t$*;*
3. пусть в графе переходов $TG_{\Pi_2}(y)$ *содержится такой ориентированный путь из вершины* $q \in R$ *в вершину* $q' \in R$*, что в нем нет промежуточных вершин из* R*, и пусть* $t = g(q, z)$*,* $t' = g(q', z)$ *— соответствующие решения на входе* z*. Тогда* $t = t'$ *или граф переходов* $TG_{\Pi_1}(z)$ *содержит дугу, исходящую из вершины* t *и входящую в вершину* t'*.*

Рассмотрим произвольный путь Q в графе $TG_{\Pi_2}(y)$ с начальной и конечной вершинами из множества R. Возьмем образ вершин этого пути в графе

$TG_{\Pi_1}(z)$ относительно отображения g. Условие 3 гарантирует, что участку пути Q между любыми двумя вершинами $q, q' \in R$ (не содержащему промежуточных вершин из R) соответствует либо вершина, являющаяся образом вершин q и q', либо ребро, соединяющее образы этих вершин. Следовательно, пути Q в графе $TG_{\Pi_1}(z)$ можно поставить в соответствие путь $Q(g)$, вершины которого являются образами вершин из Q. При этом длина пути $Q(g)$ не превосходит длины исходного пути Q.

Пусть t — произвольная вершина в графе $TG_{\Pi_1}(z)$ и пусть $q = g^{-1}(t)$ — ее прообраз из множества R. Рассмотрим один из путей минимальной длины, ведущий из вершины q в некоторый сток графа $TG_{\Pi_2}(y)$. В графе $TG_{\Pi_1}(z)$ этому пути соответствует путь небо́льшей длины, ведущий из вершины t в один из стоков этого графа. Таким образом, высота вершины q не меньше высоты вершины t в графе $TG_{\Pi_1}(z)$. Следовательно, высота графа $TG_{\Pi_2}(y)$ не меньше высоты графа $TG_{\Pi_1}(z)$. Поэтому плотная сводимость сохраняет нижние оценки на число шагов алгоритма локального спуска.

Заметим, что в условии 3 в качестве ориентированного пути может рассматриваться дуга из q в q'. В этом случае в графе $TG_{\Pi_1}(z)$ такой дуге соответствует либо вершина t, либо дуга (t, t'). Такая ситуация может возникать, если в качестве множества R выбирается, например, всё множество допустимых решений.

Определение 11 *[75] Задача* Π *из класса PLS называется плотно полной, если все задачи из этого класса плотно сводятся к ней.*

Все PLS-полные задачи, изображенные на рис.2.1 являются плотно PLS-полными [75]. Чтобы доказать, что в худшем случае алгоритму локального спуска требуется экспоненциальное число шагов для плотно PLS-полных задач, достаточно найти в классе PLS хотя бы одну задачу, обладающую этим свойством.

Лемма 4 *[75] 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 z длины n множество допустимых решений задачи $Sol(z)$ 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 $TG_{\Pi}(z)$ является путем длины $2^n - 1$. Алгоритм локального спуска, начиная с вершины $i = 2^n - 1$ выполнит экспоненциальное число шагов. Таким образом, в худшем случае для любой плотно PLS-полной задачи алгоритму локального спуска потребуется экспоненциальное число шагов для достижения локального минимума. Т. к. задача МПК с окрестностями N_{FM_1} и $N_{\textit{1-замена}}$ является плотно PLS-полной [13], то получаем следующее.

Теорема 3 *В худшем случае стандартному алгоритму локального спуска требуется экспоненциальное число шагов для достижения локального минимума в задаче МПК с окрестностями N_1 и N_{FM_1} при любом правиле замещения.*

Более сложной является задача локального поиска с фиксированной начальной точкой: для заданных входа, окрестности и начального решения найти локальный минимум, который может быть получен стандартным алгоритмом локального спуска из данного начального решения. Раньше требовалось найти любой локальный минимум, теперь надо найти локальный минимум, достижимый из заданного начального решения. Задача отыскания локального минимума с фиксированной начальной точкой лежит в классе $PSPACE$ [75]. 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.

Теорема 4 *Нахождение локального минимума в задаче МПК при фиксированной начальной точке является PSPACE-полной задачей с любой из следующих окрестностей: N_1, N_{LK}, N_{LK_1}, N_{FM}, N_{FM_1}.*

Эта теорема является следствием аналогичного утверждения для задачи о p-медиане [13].

2.3 Вычислительная сложность алгоритма локального поиска в среднем случае

Итак, в худшем случае вычислительная сложность алгоритмов основанных на локальном спуске оказывается экспоненциальной. Однако экспериментальные исследования и многочисленные приложения показывают, что с практической точки зрения найти локальный минимум для полиномиально проверяемых окрестностей не представляет труда. Возникает вопрос, является ли в среднем алгоритм локального спуска полиномиальным?

Первые результаты исследований в этом направлении могут быть найдены в [71]. В этой работе рассматривается задача максимизации вещественнозначной функции F, определенной на множестве вершин n-мерного булева куба. Предполагается, что все значения, принимаемые функцией F, различные. При таких условиях на функцию F список всех вершин можно упорядочить по значению функции от наилучшего к наихудшему. Такой порядок вершин обозначим через ν. Вершины n-мерного булева куба являются соседями по окрестности N_{Flip}, если они отличаются друг от друга только одной компонентой.

Теорема 5 *[71] В предположении, что любой порядок вершин является равновероятным, математическое ожидание числа шагов стандартного алгоритма локального поиска по окрестности N_{Flip} при любом правиле замещения не превосходит величины $\frac{3}{2}en$, где e— логарифмическая константа.*

Предположение о равновероятности появления порядков может показаться обременительным. Тем не менее утверждение остается верным и в более общем случае.

Теорема 6 *[71] Предположим, что вероятность появления порядков удовлетворяет отношению:*

$$\frac{P(\nu)}{P(\nu')} \leq 2^{\alpha n}, \quad \alpha > 0.$$

Тогда математическое ожидание числа шагов алгоритма локального спуска с окрестностью N_{Flip} не превосходит величины $(\alpha + 2)en$, где e— логарифмическая константа.

Эти результаты показывают кардинальное отличие поведения алгоритма локального поиска в худшем случае от поведения в среднем. Такой же вывод следует и из экспериментальных исследований. К сожалению, для задач размещения, в частности, для задачи МПК пока не удается найти нетривиальные распределения для величин $c_{ij},\ \ g_{ij},\ \ i \in I,\ \ j \in J$, которые удовлетворяли бы условиям теорем 5, 6. Вопрос о получении верхней оценки на среднее число шагов алгоритма для задач размещения с различными полиномиальными окрестностями остается открытым.

2.4 Погрешность локальных оптимумов

Раздел посвящен известным теоретическим результатам, касающихся качества локальных оптимумов, как приближенных решений оптимизационной задачи. Результаты верны для задачи МПК, т. к. задача о p-медиане является её частным случаем.

Определение 12 *Окрестность называется точной, если любой локальный оптимум является глобальным.*

Существование точной полиномиально проверяемой окрестности делает локальный поиск точным методом. Однако для многих задач (задачи коммивояжера, задачи о p-медиане) существование такой окрестности влечет P=NP [14].

Определение 13 *[9] Пусть $Max(z)$ – наибольшее по модулю число, содержащееся во входе z. Оптимизационная задача OP называется псевдо-полиномиально ограниченной, если существует полином p от длины входа z и число $Max(z)$ такие, что для любого входа $z \in \mathcal{I}$ и любого решения $s \in Sol(z)$ выполняется неравенство*

$$f(s, z) - f^*(z) \leq p(|z|, Max(z)),$$

где $f^*(z)$ – оптимальное значение целевой функции.

Обозначим через NPO_B множество псевдо-полиномиально ограниченных оптимизационных задач.

Теорема 7 *[13] Пусть $OP \in NPO_B$, $(OP, N) \in PLS$ и целевая функция принимает только целочисленные значения. Если P$\neq$NP и задача вычисления приближённого решения с гарантированной оценкой точности ρ для задачи $OP \in NPO_B$ является NP-трудной в сильном смысле, то найдутся исходные данные, для которых существует локальный оптимум, отклоняющийся более чем в ρ раз от оптимального решения задачи.*

Следствие 1 *Если $P \neq NP$ и задача $OP \in NPO_B$ является NP-трудной в сильном смысле, то для неё не существует точных полиномиально проверяемых окрестностей.*

Результаты теоремы 7 могут быть усилены.

Теорема 8 *[13] 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 значения оптимального решения задачи.*

Следствие 2 *Если $NP \neq$co-NP и задача OP является NP-трудной, то для неё не существует точных полиномиально проверяемых окрестностей.*

2.5 Локально седловые точки

Пусть в задаче МПК для каждого $i \in I$ заданы f_i – стоимости открытия i-го предприятия, а ограничение (1.2) отсутствует. Тогда полученная модель представляет двухуровневую задачу размещения с предпочтениями клиентов:

$$\min_y \Big(\sum_{i \in I} f_i y_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij}(y)\Big) \tag{2.1}$$

$$y_i \in \{0, 1\}, \quad i \in I, \tag{2.2}$$

где $x^*_{ij}(y)$ – оптимальное решение задачи клиента:

$$\min_X \sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} \tag{2.3}$$

$$\sum_{i\in I} x_{ij} = 1, \quad j \in J, \tag{2.4}$$

$$x_{ij} \le y_i, \quad i \in I, \quad j \in J, \tag{2.5}$$

$$x_{ij} \in \{0,1\}, \quad i \in I, \quad j \in J. \tag{2.6}$$

В [7] установлено сведение этой задачи к задачи минимизации полиномов от булевых переменных. Впервые независимо друг от друга такое сведение было предложено Бересневым в [3] и Хаммером в [41] для простейшей задачи размещения, которая является частным случаем задачи (2.1)-(2.6), когда матрицы (c_{ij}) и (g_{ij}) совпадают. Аналогичное сведение легко построить и для задачи МПК. Для этого упорядочим каждый столбец $j \in J$ по возрастанию:

$$g_{i_1^j j} < g_{i_2^j j} < \ldots < g_{i_m^j j}$$

и $S_{ij} = \{k \in I | g_{kj} < g_{ij}\}, i \in I, j \in J$. Обозначим $\nabla c_{i_1^j j} = c_{i_1^j j}$, $\nabla c_{i_l^j j} = c_{i_l^j j} - c_{i_{l-1}^j j}$, $1 < l \le m$, $z_i = 1 - y_i, i \in I, j \in J$. Выпишем следующую задачу.

Минимизировать значение полинома:

$$P(z) = \sum_{i\in I}\sum_{j\in J} \nabla c_{ij} \prod_{k\in S_{ij}} z_k \tag{2.7}$$

при ограничениях:

$$\sum_{i\in I} z_i = m - p, \tag{2.8}$$

$$z_i \in \{0,1\}, i \in I. \tag{2.9}$$

Теорема 9 *[2] Задача (1.1)-(1.7) эквивалентна задаче минимизации полинома от булевых переменных (2.7)-(2.9).*

Заменим условие целочисленности (2.9) на непрерывное условие принадлежности отрезку $[0,1]$ и выпишем для релаксированной задачи (2.7)-(2.9) 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-Таккера имеют вид:

$$\frac{\partial L}{\partial z_i}(z, \lambda, \mu, \sigma) = P_i'(z) - \lambda + \sigma_i - \mu_i = 0, i \in I, \tag{2.10}$$

$$\sum_{i \in I} z_i = m - p, \tag{2.11}$$

$$0 \le z_i \le 1, i \in I \tag{2.12}$$

$$\sigma_i(z_i - 1) = 0, i \in I, \tag{2.13}$$

$$\mu_i z_i = 0, i \in I. \tag{2.14}$$

Определение 14 *Вектор $(z^*, \lambda^*, \mu^*, \sigma^*)$ называется локально седловой точкой относительно окрестности N_1, 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 вектора из окрестности $N_1(z^*)$.

Теорема 10 *Для любого допустимого решения Y^* задачи МПК 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 локально седловой точкой относительно окрестности N_1 функции L;
2) Y^* является локальным оптимумом относительно окрестности N_1;
3) $z(Y^*)$ удовлетворяет условиям Куна-Таккера.

Доказательство. Проверим, что из 1) следует 2). Пусть $(z(Y^*), \lambda^*, \mu^*, \sigma^*)$ седловая точка относительно окрестности N_1. Обозначим $z^* = z(Y^*)$. Из неравенства (1) получаем, что

$$L(z^*, \lambda^*, \mu^*, \sigma^*) \overset{(3)}{=} \sup_{\lambda, \mu \ge 0, \sigma \ge 0} L(z^*, \lambda, \mu, \sigma) \overset{(4)}{=} P(z^*).$$

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) не верно. 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

$$P(z^*) \le L(z, \lambda^*, \mu^*, \sigma^*) \text{ для всех } z \in N_1(z^*).$$

Так как любой вектор $z \in N_1(z^*)$ 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 $Y(z) \in N_1(Y^*)$.
Следовательно, Y^* является локальным оптимумом относительно окрестности N_1.
Проверим, что из 2) следует 3). Пусть Y^* локальный оптимум относительно окрестности N_1. Тогда булев вектор $z^* = z(Y^*)$ удовлетворяет ограничению $\sum_{i\in I} z_i^* = m - p$ и является оптимальным решением для $P(z)$ по окрестности N_1. 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 N_1 функции L. Пусть $P''_{i_0 i_1}(z)$ вторая частная производная $P(z)$ 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 $z \in N_1(z^*)$, $z^*_{i_0} = 0$, $z^*_{i_1} = 1$, $z_{i_0} = 1$, $z_{i_1} = 0$ 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

$$\lambda^*(\sum_{i \in I} z_i^* - m + p) = 0, \sigma_i^*(z_i^* - 1) = 0, \mu_i^* z_i^* = 0, i \in I.$$

А также

$$\frac{\partial L}{\partial z_i}(z^*, \lambda^*, \mu^*, \sigma^*) = P^{'}_i(z^*) - \lambda^* + \sigma_i^* - \mu_i^* = 0,\ i = 1, \ldots, m.$$

Доказали, что из 2) следует 3).
Проверим, что из 3) следует 1). Из условий дополняющей нежесткости получаем

$$L(z^*, \lambda^*, \mu^*, \sigma^*) = P(z^*).$$

Возьмём $z \in N_1(z^*)$, $z^*_{i_0} = 0$, $z^*_{i_1} = 1$, $z_{i_0} = 1$, $z_{i_1} = 0$ 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 $\sigma^*_{i_1} = \lambda^* - P^{'}_{i_1}(z^*)$, $\mu^*_{i_0} = P^{'}_{i_0}(z^*) - \lambda^*$. Из этого следует, что

$$L(z, \lambda^*, \mu^*, \sigma^*) = P(z) + P^{'}_{i_1}(z^*) - P^{'}_{i_0}(z^*).$$

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

$$L(z^*,\lambda,\mu,\sigma) = P(z^*) + \lambda(\sum_{i\in I} z_i^* - m + p) + \sum_{i\in I}\sigma_i(z_i^*-1) - \sum_{i\in I}\mu_i z_i^* =$$

$$= P(z^*) - \sum_{i:\ z_i^*=0}\sigma_i - \sum_{i:\ z_i^*=1}\mu_i \leq P(z^*) = L(z^*,\lambda^*,\mu^*,\sigma^*).$$

Теорема доказана.

Глава 3

Генетические алгоритмы

В данной главе для решения рассматриваемой задачи предлагается алгоритм генетического локального поиска, использующий в качестве популяции локальные оптимумы по окрестности Лина–Кернигана. Для оценки качества получаемых решений используются сведéния исходной задачи к задачам целочисленного линейного программирования. Предложено новое сведéние, доминирующее уже известные по значению целевой функции линейной релаксации. Проведены численные эксперименты на примерах с большим разрывом двойственности.

3.1 Нижние оценки оптимального решения

Рассмотрим несколько способов сведéния задачи (1.1)–(1.7) к задачам ЦЛП и покажем их различия при вычислении нижних оценок методами линейного программирования.

3.1.1 Сведéния к задачам ЦЛП

Определим для каждого столбца $j \in J$ матрицы (g_{ij}) перестановку $(i_1^j, ..., i_m^j)$:

$$g_{i_1^j j} < g_{i_2^j j} < \cdots < g_{i_m^j j} \tag{3.1}$$

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:

$$(x_{ij} = 1) \Longrightarrow (y_k = 0),\ k \in S_{ij}. \tag{3.2}$$

Тогда задача (1.1)–(1.7) может быть переписана в виде: найти

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{3.3}$$

при ограничениях:

$$y_k \leq 1 - x_{ij}, \quad k \in S_{ij}; \quad i \in I; j \in J \tag{3.4}$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J \tag{3.5}$$

$$\sum_{i \in I} y_i = p, \tag{3.6}$$

$$x_{ij} \leq y_i, \; i \in I; j \in J \tag{3.7}$$

$$x_{ij}, y_i \in \{0, 1\}, \quad i \in I. \tag{3.8}$$

Действительно, для оптимального решения задачи (3.3)–(3.8) все ограничения исходной задачи (1.1)–(1.7) будут выполнены, а группа ограничений (3.4) 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 сведéнии число переменных осталось прежним и равным $m + mn$, а число ограничений стало на $O(m^2 n)$ больше. Чтобы избежать большого числа дополнительных ограничений, условие (3.2) можно переписать в виде:

$$\sum_{k \in S_{ij}} y_k \leq |S_{ij}|(1 - x_{ij}), \quad i \in I; j \in J. \tag{3.9}$$

Эти ограничения (3.9) получаются суммированием ограничений (3.4). Новое сведéние приводит аналогичным образом к новой нижней оценке. Обозначим её LB_2. Число дополнительных ограничений сократилось до mn, оптимальное решение задачи от этого не изменилось, но линейная релаксация стала более слабой, $LB_1 \geq LB_2$. Условие (3.2) можно переписать также в виде:

$$y_i \leq x_{ij} + \sum_{k \in S_{ij}} y_k \quad i \in I; j \in J. \tag{3.10}$$

что даёт еще одно сведéние и нижнюю оценку LB_3. Приведенные нижние оценки были предложены в работе [6]. В данной работе рассматривается новое сведéние к задаче ЦЛП.

Лемма 5 *Условия (3.4), (3.5), (3.7) определяют те же величины* (x_{ij}), (y_i), *что (3.5), (3.7) и*

$$y_i \le x_{ij} + \sum_{k \in S_{ij}} x_{kj} \quad i \in I; j \in J. \tag{3.11}$$

Доказательство. Пусть решение $(x_{ij}), (y_i)$ удовлетворяет условиям (3.4), (3.5), (3.7). Покажем, что верно неравенство (3.11). Т. к.

$$y_k \le 1 - x_{ij} \;\; k \in S_{ij}, i \in I, j \in J,$$

тогда, если $y_k = 1$ для некоторого $k \in S_{ij}$, то $x_{ij} = 0$ для всех $i \in T_{kj}, j \in J$. Следовательно, учитывая $\sum_{i \in I} x_{ij} = 1, j \in J$, получаем, что $x_{kj} + \sum_{i \in S_{kj}} x_{ij} = 1$, а значит $y_k \le x_{kj} + \sum_{i \in S_{kj}} x_{ij}$ для каждого $i \in I, j \in J$ и (3.11) верно.

Пусть $(x_{ij}), (y_i)$ удовлетворяет условиям (3.5), (3.7) и (3.11). Тогда в силу ограничений (3.7) и (3.11) верно (3.10). Лемма 1 доказана.

Покажем, что получаемая с помощью (3.11) новая нижняя оценка, назовём ее LB_4, доминирует три предшествующие по значению целевой функции линейной релаксации.

Теорема 11 $LB_4 \ge \max(LB_1, LB_2, LB_3)$.

Доказательство. Перепишем ограничение (3.5) в виде

$$1 - \sum_{k \in T_{ij}} x_{kj} = x_{ij} + \sum_{k \in S_{ij}} x_{kj}.$$

Тогда ограничение (3.11) примет вид $y_i \le 1 - \sum\limits_{k \in T_{ij}} x_{kj}$, следовательно, будет выполнено (3.4). Ограничения (3.9) остаются верными, т. к. получаются суммированием ограничений (3.4). Поскольку $x_{ij} \le y_i, \;\; i \in I, j \in J$, то

$$x_{ij} + \sum_{k \in S_{ij}} x_{kj} \le x_{ij} + \sum_{k \in S_{ij}} y_k,$$

следовательно, (3.10) выполняется. Таким образом, значение LB_4 не может быть меньше LB_1, LB_2, LB_3. Теорема 11 доказана.

3.1.2 Сведéние к задаче с парой матриц

Рассмотрим матрицы $A = (a_{ij}), i \in I, j \in J_1$ и $B = (b_{ij}), i \in I, j \in J_2$, у которых число строк совпадает, а число столбцов может быть различным. Задача с парой матриц (ПМ) заключается в том, чтобы найти непустое подмножество строк $S \subseteq I$ на котором достигается минимум следующей целевой функции [1]:

$$R(S) = \sum_{j \in J_1} \max_{i \in S} a_{ij} + \sum_{j \in J_2} \min_{i \in S} b_{ij}.$$

Заметим, что если $J_1 = I$, $a_{ij} = a_i$ при $i = j$ и $a_{ij} = 0$, в противном случае, то получаем простейшую задачу размещения [54]. Таким образом, задача с парой матриц является NP-трудной в сильном смысле.

Известно [8], что для задачи (1.1)–(1.7) можно построить сведéние к задаче ПМ с дополнительным ограничением $|S| = p$. Для построения сведéния представим матрицу (c_{ij}) в виде суммы двух матриц $c_{ij} = a_{ij} + b_{ij}, \quad i \in I, j \in J$, где

$$a_{i_1^j j} = 0, \quad a_{i_k^j j} = \sum_{l=1}^{k-1} \min\{0; c_{i_{l+1}^j j} - c_{i_l^j j}\}, k = 2, \ldots, m, j \in J, \tag{3.12}$$

$$b_{i_1^j j} = c_{i_1^j j}, \quad b_{i_k^j j} = c_{i_1^j j} + \sum_{l=1}^{k-1} \max\{0; c_{i_{l+1}^j j} - c_{i_l^j j}\}, k = 2, \ldots, m, j \in J, \tag{3.13}$$

перестановка $(i_1^j, \ldots, i_m^j), j \in J$ соответствует (3.1). Очевидно, что

$$a_{i_1^j j} \geq a_{i_2^j j} \geq \cdots \geq a_{i_m^j j}$$

$$b_{i_1^j j} \leq b_{i_2^j j} \leq \cdots \leq b_{i_m^j j}$$

при всех $j \in J$.

Тогда, задача (1.1)–(1.7) 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 виде задачи 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 представления для задачи ПМ, 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 представим задачу ПМ в следующем виде: найти

$$\min\{\sum_{i\in I}\sum_{j\in J} a_{ij}z_{ij} + \sum_{i\in I}\sum_{j\in J} b_{ij}x_{ij}\} \tag{3.14}$$

при ограничениях:

$$y_i \leq \sum_{k\in S_{ij}} z_{kj} + z_{ij}, \quad i \in I; j \in J; \tag{3.15}$$

$$\sum_{i\in I} x_{ij} = 1, \qquad j \in J \tag{3.16}$$

$$\sum_{i\in I} z_{ij} = 1, \qquad j \in J \tag{3.17}$$

$$\sum_{i\in I} y_i = p, \tag{3.18}$$

$$0 \leq x_{ij} \leq y_i, \quad i \in I; j \in J \tag{3.19}$$

$$0 \leq z_{ij} \leq y_i, \quad i \in I; j \in J \tag{3.20}$$

$$z_{ij}, x_{ij}, y_i \in \{0,1\}, \quad i \in I; j \in J. \tag{3.21}$$

Поскольку $b_{i_1^j j} \leq b_{i_2^j j} \leq \dots \leq b_{i_m^j j}, j \in J$, то каждый столбец j матрицы (b_{ij}) задает тот же порядок, что и столбец j матрицы (g_{ij}). Ограничения (3.15) гарантируют, что j-й клиент делает выбор согласно собственным предпочтениям. Обозначим через LB_5 оптимальное значение линейной релаксации задачи (3.14)–(3.21), а через LB_5' оптимальное значение линейной релаксации этой же задачи с дополнительным набором ограничений $x_{ij} = z_{ij}, i \in I,\ j \in J$. Заметим, что $LB_5' \geq LB_5$ и $LB_5' = LB_4$.

Теорема 12 *Для любого $N > 0$ существуют такие исходные данные задачи (1.1)–(1.7), что $LB_5 \leq -N$.*

Доказательство. Для целого $k \geq N+2$ полагаем $I = \{1,\dots,k+p\}$, $J =$

$\{1,2\}, p \in Z^+$, $d_{ij} = k + p - i$ и

$$c_{ij} = \begin{cases} 0, & \text{если} \quad i < p, \quad j = 1; \\ k, & \text{если} \quad i = p, \quad j = 1; \\ 1, & \text{если} \quad i > p, \quad j = 1; \\ 1, & \text{если} \quad i < p, \quad j = 2; \\ 0, & \text{если} \quad i \geq p, \quad j = 2; \end{cases} \qquad i \in I, j \in J.$$

Согласно (3.12), (3.13) получаем

$$a_{ij} = \begin{cases} -k, & \text{если} \quad i < p, \quad j = 1; \\ 0, & \text{если} \quad i \geq p, \quad j = 1; \\ 0, & \text{если} \quad j = 2; \end{cases} \qquad b_{ij} = \begin{cases} k, & \text{если} \quad i \leq p, \quad j = 1; \\ 1, & \text{если} \quad i > p, \quad j = 1; \\ c_{ij}, & \text{если} \quad j = 2. \end{cases}$$

Решение

$$x_{ij} = \begin{cases} 0, & \text{если} \quad i \leq p, \quad j \in J; \\ 1/k, & \text{если} \quad i > p, \quad j \in J; \end{cases} \qquad y_i = \begin{cases} 1, & \text{если} \quad i < p; \\ 0, & \text{если} \quad i = p; \\ 1/k, & \text{если} \quad i > p; \end{cases}$$

$$z_{ij} = \begin{cases} 1 - 1/k, & \text{если} \quad i = p - 1, \quad j = 1; \\ 1/k, & \text{если} \quad i = p + k, \quad j = 1; \\ 0, & \text{если} \quad i \neq p - 1, \quad i \neq p + k, \quad j = 1; \\ 0, & \text{если} \quad i \leq p, \quad j = 2; \\ 1/k, & \text{если} \quad i > p, \quad j = 2; \end{cases}$$

является допустимым решением релаксированной задачи (3.14)–(3.20) и $LB_5 \leq (-k)(1 - 1/k) + \sum\limits_{i=1}^{k} 1/k = 2 - k$. Теорема 12 доказана.

3.1.3 Новое сведéние к задаче с парой матриц

Представим новую формулировку задачи ПМ 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 $(i^j_1, ..., i^j_m)$

$$\Delta^j_l = \min\{0; c_{i^j_{l+1} j} - c_{i^j_l j}\}, l = 1, \ldots, m - 1.$$

Пусть $L_j = \{l \in \{1, ..., m - 1\} \mid \Delta^j_l < 0\}$. Заметим, что при заданном $j \in J$ по номеру $l \in L_j$ однозначно восстанавливается номер $i^j_l \in I$. Заменим j-й столбец матрицы (a_{ij}) на $|L_j|$ столбцов вида

$$\overline{a}_{il} = \begin{cases} 0, & \text{если } \ i \in T_{i^j_l j} \\ -\Delta^j_l, & \text{если } \ i \notin T_{i^j_l j} \end{cases} \ i \in I, \ l \in L_j.$$

Тогда

$$a_{ij} = \sum_{l \in L_j} (\overline{a}_{il} + \Delta^j_l), i \in I, j \in J$$

и задача ПМ может быть переписана в виде:

$$\min_{y_i \in \{0,1\}, \sum\limits_{i \in I} y_i = p} \{ \sum_{j \in J} \sum_{l \in L_j} \max_{i | y_i = 1} \overline{a}_{il} + \sum_{j \in J} \min_{i | y_i = 1} b_{ij} \} + \sum_{j \in J} \sum_{l \in L_j} \Delta^j_l. \tag{3.22}$$

Введём дополнительные переменные $v^j_l \in \{0, 1\}, \ l \in L_j, \ j \in J$ и представим эту задачу следующим образом: найти

$$\min \{ \sum_{j \in J} \sum_{l \in L_j} -\Delta^j_l v^j_l + \sum_{i \in I} \sum_{j \in J} b_{ij} x_{ij} \} + \sum_{j \in J} \sum_{l \in L_j} \Delta^j_l \tag{3.23}$$

при ограничениях:

$$y_i \leq x_{ij} + \sum_{k \in S_{ij}} x_{kj}, \ i \in I, \ j \in J, \tag{3.24}$$

$$\sum_{i \in I} x_{ij} = 1, \ j \in J, \tag{3.25}$$

$$\sum_{i \in I} y_i = p, \tag{3.26}$$

$$0 \leq x_{ij} \leq y_i, \ i \in I, \ j \in J, \tag{3.27}$$

$$v^j_l \geq \sum_{k \notin T_{i^j_l j}} x_{kj'}, \ l \in L_j, \ j \in J, \ j' \in J, \tag{3.28}$$

$$v^j_l, y_i, x_{ij} \in \{0, 1\}, \ l \in L_j, \ j \in J, \ i \in I. \tag{3.29}$$

Обозначим через LB_6 оптимальное значение линейной релаксации этой задачи.

Теорема 13 $LB_6 \geq LB_4$.

Доказательство. Покажем, что задача (3.3), (3.11), (3.5)–(3.8) эквивалентна задаче (3.23)–(3.27), (3.29) с дополнительным ограничением

$$v_l^j = \sum_{k \notin T_{i_l^j j}} x_{kj}, \ l \in L_j, \ j \in J. \tag{3.30}$$

Так как $c_{ij} = a_{ij} + b_{ij}, \ \ i \in I, j \in J$, 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 (3.23)–(3.27), (3.29), (3.30). Действительно,

$$\sum_{l \in L_j} -\Delta_l^j v_l^j + \sum_{l \in L_j} \Delta_l^j = \sum_{l \in L_j} \Delta_l^j (1 - v_l^j)$$

$$= \sum_{l \in L_j} \Delta_l^j \sum_{k \in T_{i_l^j j}} x_{kj} = \sum_{k=2}^{m} x_{i_k^j j} \sum_{l=1}^{k-1} \Delta_l^j$$

$$= \sum_{k=2}^{m} a_{i_k^j j} x_{i_k^j j} = \sum_{i \in I} a_{ij} x_{ij}.$$

Заметим, что это равенство верно как для целочисленных значений переменных, так и для дробных. Следовательно, задача (3.3), (3.11), (3.5)–(3.8) эквивалентна задаче (3.23)–(3.27), (3.29), (3.30). Тогда значение LB_4 равно оптимальному значению линейной релаксации задачи (3.23)–(3.27), (3.30). Причем, если ограничения (3.30) заменить на ограничения

$$v_l^j \geq \sum_{k \notin T_{i_l^j j}} x_{kj}, \ l \in L_j, \ j \in J, \tag{3.31}$$

то оптимальное значение не изменится, так как рассматриваемая задача на минимум, коэффициенты в целевой функции (3.23) при v_l^j положительные ($-\Delta_l^j \geq 0$) и, следовательно, значения v_l^j однозначно определяются по переменным x_{ij}. Но ограничения (3.31) составляют часть ограничений (3.28), следовательно $LB_4 \leq LB_6$. Теорема 13 доказана.

Теорема 14 *Для любого $N > 0$ существуют такие исходные данные задачи (1.1)–(1.7), что $LB_6/LB_4 \geq N$.*

Доказательство. Пусть $k = N$ и исходные данные те же, что и в доказательстве теоремы 12. 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 (3.3), (3.5)–(3.7), (3.11). Следовательно, $LB_4 \leq 1/k$. Двойственная задача к задаче (3.23)– (3.24) имеет вид: найти

$$\max(\sum_{j \in J} u_j + pr) + \sum_{j \in J} \sum_{l \in L_j} \Delta_l^j$$

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: $L_j = \{1\}, \Delta_l^j = -k$ для $j = 1, L_j = \emptyset$, для $j = 2$ и

$$\bar{a}_{ij} = \begin{cases} 0, & \text{если} \quad i < p, \quad j = 1; \\ k, & \text{если} \quad i \geq p \quad j = 1; \end{cases}$$

$$b_{ij} = \begin{cases} k, & \text{если} \quad i \leq p, \quad j = 1; \\ 1, & \text{если} \quad i > p, \quad j = 1; \\ 1, & \text{если} \quad i < p, \quad j = 2; \\ 0, & \text{если} \quad i \geq p, \quad j = 2. \end{cases}$$

Решение

$$w_{ij} = \alpha_{ij} = r = 0, u = (k, 1), t_{lj'} = \begin{cases} k-1, & \text{если} \quad l = 1, \ \ j' = 1; \\ 1, & \text{если} \quad l = 1, \ \ j' = 2; \end{cases}$$

является допустимым в двойственной задаче. Следовательно, $LB_6 \geq k + 1 - k = 1$. Теорема 14 доказана.

3.2 Процедура Резенде и Вернека

На каждом шаге процедуры локального поиска необходимо просматривать окрестность текущего решения. Рассмотрим окрестность $N_1(y)$, описание которой приводилось в первой главе. Обозначим через S множество номеров открытых предприятий в решении y, т. е. $S = \{i \in I | y_i = 1\}$. Тогда $N_1(y)$ содержит все решения, полученные при открытии предприятия не принадлежащего S и закрытии некоторого предприятия из S. Таким образом, при просмотре окрестности необходимо перебирать всевозможные пары $(i_S, i_{\overline{S}})$, где $i_S \in S, i_{\overline{S}} \in I \setminus S$ и вычислять значение целевой функции на новом решении. Если в качестве правила замещения выбран спуск в направлении наилучшего, наихудшего или случайного элемента, то перебирать приходится все возможные парные замены. Тейтцом и Бартом в работе [72] для классической задачи о p-медиане была предложена процедура, которая позволяет выполнить спуск в направление наилучшего элемента с трудоемкостью $O(mnp)$. Резенде и Вернек предложили схему [66], которая позволяет существенно ускорить этот процесс до $O(\max(p, m)n)$. Эта схема может быть адаптирована к задаче МПК следующим образом. Введем вспомогательные обозначения:

$\Phi_1(j), j \in J$ – номер предприятия, которому j-ый клиент отдает наибольшее предпочтение;

$\Phi_2(j), j \in J$ – номер следующего по предпочтениям после $\Phi_1(j)$ предприятия;

$g_1(j), g_2(j)$ – величины $g_{\Phi_1(j)j}, g_{\Phi_2(j)j}$, соответственно.

$c_1(j), c_2(j)$ –транспортные расходы при обслуживании j-го клиента из предприятий $\Phi_1(j), \Phi_2(j)$, соответственно.

Вычислим для каждого $i_{\overline{S}} \in I \setminus S$ на сколько изменятся транспортные расходы, если открытие $(p+1)$-го предприятия в интересах j-го клиента, т. е.

$g_{i_{\overline{S}}j} < g_1(j)$:

$$gain(i_{\overline{S}}) = \sum_{j \in J | g_{i_{\overline{S}}j} < g_1(j)} (c_1(j) - c_{i_{\overline{S}}j}).$$

Вычислим для каждого $i_S \in S$ на сколько изменятся транспортные расходы при закрытии предприятия $\Phi_1(j)$:

$$loss(i_S) = \sum_{j \in J | \Phi_1(j) = i_S} (c_2(j) - c_1(j)).$$

Величину $profit$ определим следующим образом:

$$profit(i_S, i_{\overline{S}}) = gain(i_{\overline{S}}) - loss(i_S) + extra(i_{\overline{S}}, i_S).$$

Чтобы вычислить величину $extra(i_{\overline{S}}, i_S)$ рассмотрим возможные случаи при нахождении пары $(i_{\overline{S}}, i_S)$.

1.$\Phi_1(j) \neq i_S$

В этом случае $loss(i_S)$ равно нулю, т. к. прежнее наилучшее с точки зрения j-го клиента предприятие $\Phi_1(j)$ осталось в решении. Следовательно, может измениться только j-ое слагаемое в $gain(i_{\overline{S}})$.

2.$\Phi_1(j) = i_S$

В этом случае кандидатом на закрытие оказалось "наилучшее" для j-го клиента предприятие. Рассмотрим возможные ситуации.

2.1 $g_1(j) \leq g_{i_{\overline{S}}j} < g_2(j)$, тогда j-ый клиент отдаст предпочтение предприятию $i_{\overline{S}}$ и транспортные расходы изменятся на величину $(c_{i_{\overline{S}}j} - c_1(j))$, а не $(c_2(j) - c_1(j))$, следовательно, величину $loss(i_S)$ нужно скорректировать, т. е. вычесть $(c_2(j) - c_{i_{\overline{S}}j})$, это и будет одно из слагаемых в $extra$.

2.2 $g_{i_{\overline{S}}j} \leq g_1(j)$. В этом случае j-ый клиент отдаст предпочтение предприятию $i_{\overline{S}}$, следовательно, изменение в транспортных расходах будет учтено в $gain(i_{\overline{S}})$, поэтому нужно уничтожить соответствующее j-ое слагаемое в $loss(i_S)$, т. е. вычесть величину $(c_2(j) - c_1(j))$.

Итак, с учетом 2.1 и 2.2 величина $extra(i_{\overline{S}}, i_S)$ вычисляется так

$$extra(i_{\overline{S}}, i_S) = \sum_{j \in J | (\Phi_1(j) = i_{\overline{S}}) \wedge (g_1(j) \leq g_{i_{\overline{S}}j} \leq g_2(j))} (c_2(j) - c_{i_{\overline{S}}j}) +$$

$$\sum_{j \in J | (\Phi_1(j) = i_{\overline{S}}) \wedge (g_{i_{\overline{S}}j} < g_1(j))} (c_2(j) - c_1(j)).$$

Используя введенные обозначения алгоритм локального спуска по окрестности N_1 может быть записан в виде.

Алгоритм локального спуска$(S, \Phi_1, \Phi 2)$

1. Положить $A := J$
2. Инициализировать величины $gain(i) := 0, loss(i) := 0$, $extra(j, i) := 0$, $i \in I, j \in \widetilde{S}(\widetilde{S} \subset I, |\widetilde{S}| = p)$
3. Пока не найден локальный минимум выполнять:
   3.1 для каждого $j \in A$ вызвать процедуру *П1*$(j, gain, loss, extra, \Phi_1(j), \Phi_2(j))$
   3.2 найти пару $(i_{\overline{S}}, i_S)$ и $profit$ с помощью процедуры *П2*$(gain, loss, extra)$
   3.3 если $profit \leq 0$, то иди на шаг 4;
   3.4 положить $A := \emptyset$
   3.5 для каждого $j \in J$ выполнить
   если $(\Phi_1(j) = i_{\overline{S}})$ или $(\Phi_2(j) = i_{\overline{S}})$ или $(g_{i_{\overline{S}}j} < g_2(j))$, то
   положить $A := A \bigcup \{j\}$
   3.6 для каждого $j \in A$ вызвать процедуру *П3*$(j, gain, loss, extra, \Phi_1(j), \Phi_2(j))$
   3.7 положить $S := S \bigcup i_{\overline{S}}$
   3.8 положить $S := S \setminus \{i_S\}$
   3.9 вызвать процедуру *П4*$(S, i_{\overline{S}}, i_S, \Phi_1, \Phi_2, F)$
4. Завершить работу алгоритма

Приведенный алгоритм начинает работу с допустимого решения y задачи (1.8)–(1.10), по которому определяется множество S и номера предприятий Φ_1, Φ_2. Для того чтобы каждый раз не пересчитывать величины $gain, loss, extra$ для всех $j \in J$ отводится вспомогательный массив A с номерами тех клиентов, которые обслуживаются предприятиями i_S и $i_{\overline{S}}$. Другими словами, элементы A – это те столбцы в матрице транспортных расходов, для которых будут пересчитываться величины $gain, loss, extra$ при подходящей замене. Процедура *П1* по данным Φ_1, Φ_2 вычисляет $gain, loss, extra$ для каждого $j \in A$.

Процедура П1$(j, loss, gain, extra, \Phi_1(j), \Phi 2(j))$

1. вычислить $loss(i_S) \leftarrow loss(i_S) + (c_2(j) - c_1(j))$;
2. для всех $(i_{\overline{S}} \in I \setminus S)$ выполнить
   2.1 если $g_{i_{\overline{S}}j} < g_1(j)$, то пересчитать $gain(i_{\overline{S}}) \leftarrow gain(i_{\overline{S}}) + (c_1(j) - c_{i_{\overline{S}}j})$
   2.2 если $g_1(j) \leq g_{i_{\overline{S}}j} < g_2(j)$, то положить $m_1 = c_2(j) - c_{i_{\overline{S}}j}$
   2.3 иначе положить $m_1 = 0$

2.4 если $g_{i_{\overline{S}}j} < g_1(j)$, то положить $m_2 = c_2(j) - c_1(j)$

2.5 иначе положить $m_2 = 0$

2.8 вычислить $extra(i_{\overline{S}}, i_S) \leftarrow extra(i_{\overline{S}}, i_S) + m_1 + m_2$

3. конец процедуры

Результатом работы процедуры *П2* будет пара $(i_S, i_{\overline{S}})$ с наибольшим значением $profit$. Эта процедура в качестве кандидата на открытие находит предприятие $i_{\overline{S}}$, у которого величина $gain(i_{\overline{S}})$ наибольшая, а в качестве кандидата на закрытие – предприятие i_S с наименьшим значением $loss(i_S)$. Для просматриваемых пар $(i_S, i_{\overline{S}})$ вычисляется величина $profit$. После того, как найдены кандидаты $i_S, i_{\overline{S}}$ проверим выгодна ли будет это замена, т. е. положительна ли величина $profit$? Если да, то массив A нужно переопределить, теперь он будет состоять только из тех номеров клиентов, на которых "повлияет" замена i_S на $i_{\overline{S}}$. Для всех $j \in A$ с помощью процедуры *П3* пересчитываются значения $gain, loss, extra$. Процедура *П3* аналогична *П1*, но с заменой в строках 1, 2.1, 2.8 знака плюс на знак минус. Процедура *П4* вычисляет значение целевой функции и для каждого j-го клиента определяет наиболее предпочтительные предприятия $\Phi_1(j), \Phi_2(j)$.

Трудоемкость алгоритма локального спуска.
Основные структуры данных, используемые в этом алгоритме – это массивы $gain, loss$ размерности m и двумерный массив $extra$ размерности $((m-p)p)$. Инициализация этих данных требует $O(pm)$ операций. На каждом шаге локального спуска выполняется несколько процедур, оценим трудоемкость каждой из них, чтобы посчитать трудоемкость всей процедуры. Вызов процедур *П1* и *П2* осуществляется для каждого $j \in A$ $(|A| \leq n)$, затем для каждого $i_{\overline{S}} \in I \setminus S$ пересчитываются $gain(i_{\overline{S}})$ и $extra(i_{\overline{S}}, i_S)$. Итак, в худшем случае трудоемкость каждой из этих процедур можно оценить величиной $O(n(m-p))$. Оценим трудоемкость нахождения наилучшей парной замены (процедура *П2*). Число всех возможных парных замен составляет $(p(m-p))$, поэтому можно вычислить для всех пар $profit$ и затем выбрать наилучший, но тоже самое можно сделать и за меньшее время. Для этого вспомним, что значение элементов в массиве $extra(i_{\overline{S}}, i_S)$ отлично от нуля, только когда именно i_S является наиболее предпочтительным для клиента, как правило, это бывает нечасто. Следовательно, можно хранить только ненулевые элементы в $extra$. В целом, эта процедура требует

$O(m + \Lambda mp)$ времени, где Λ – часть тех пар, для которых значение $extra$ ненулевое. И последний шаг алгоритма, процедура *П4*, переопределение наиболее предпочтительных клиентами предприятий и пересчёт целевой функции. Всё это можно выполнить за $O(pn)$ действий. Получаем, что итоговая трудоемкость алгоритма локального спуска оценивается величиной $O(max(n, p)m)$.

3.3 Генетические алгоритмы

3.3.1 Общая схема алгоритма

Алгоритм решения задач оптимизации, основанный на идеях наследственности в биологических популяциях, впервые был предложен Дж.Холландом (1975г.). Он получил название репродуктивного плана Холланда, и широко использовался как базовый алгоритм в эволюционных вычислениях. Дальнейшее развитие эти идеи, как и название – генетические алгоритмы, получили в работах Гольдберга и Де Йонга [40]. К решению оптимизационных задач эти нестандартные идеи впервые были применены в середине 70-х годов прошлого столетия [15, 32]. Примерно через десять лет появились первые теоретические обоснования этого подхода [45, 69]. На сегодняшний день генетические алгоритмы доказали свою конкурентоспособность при решении многих NP-трудных задач [11, 39] 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. Цель генетического алгоритма при решении задачи оптимизации состоит в том, чтобы найти лучшее возможное, но не гарантированно оптимальное решение. Общую схему генетических алгоритмов проще всего понять, рассматривая задачи безусловной оптимизации

$$\min\{F(x) \mid x \in B^n\}, \quad B^n = \{0,1\}^n.$$

Стандартный генетический алгоритм начинает свою работу с формирования начальной *популяции* $\mathcal{X}_0$ — конечного набора допустимых решений задачи. Эти решения могут быть выбраны случайным образом, получены с помощью вероятностных жадных алгоритмов или другими методами.

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.

Для оценки допустимых решений кроме целевой функции часто используется функция пригодности $\Phi(x)$. Она выбирается с учетом специфики конкретной оптимизационной задачи и должна быть чувствительна к небольшим изменениям аргумента. Целевая функция не всегда удовлетворяет этому свойству как, например, в задачах упаковки в контейнеры, задачах теории расписаний на параллельных машинах или при календарном планировании в условиях ограниченных ресурсов. Удачный выбор функции $\Phi(x)$ может существенно повысить эффективность генетических алгоритмов, хотя в простейшем случае можно предполагать, что $\Phi(x) = F(x)$.

На каждом шаге генетического алгоритма с помощью вероятностного оператора *селекции* выбираются из популяции два решения x^1, x^2 в качестве *родителей*. Оператор *скрещивания* по этим решениям строит новое решение x'. Оно подвергается небольшим случайным модификациям, которые принято называть *мутациями*. Затем решение добавляется в популяцию, а решение с наибольшим значением функции пригодности удаляется из популяции. Общая схема такого алгоритма может быть записана следующим образом.

<u>Генетический алгоритм</u>

1. Выбрать начальную популяцию $\mathcal{X}_0$ и положить
   $\Phi^* = \min\{\Phi(x) \mid x \in \mathcal{X}_0\}, k := 0.$
2. Пока не выполнен критерий остановки делать следующее.
   2.1. Выбрать родителей x^1, x^2 из популяции $\mathcal{X}_k$.
   2.2. Построить x' по решениям x^1, x^2.
   2.3. Модифицировать x'.
   2.4. Если $\Phi^* > \Phi(x')$, то положить $\Phi^* := \Phi(x')$.
   2.5. Обновить популяцию и положить $k := k + 1$.

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 x в качестве одного из родителей обратно пропорционально значению $\Phi(x)$. При турнирной селекции формируется случайное подмножество из элементов популяции и среди них выбирается один элемент с наименьшим значением функции пригодности. Турнирная селекция имеет определенные преимущества перед пропорциональной, так как не теряет своей избирательности, когда в ходе эволюции все элементы популяции становятся примерно равными по пригодности. 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, часть из которых не ограничивается двумя родителями. Однородный оператор скрещивания является, наверное, одним из наиболее простых и понятных. По решениям x^1, x^2 он строит решение x', присваивая каждой координате этого вектора с вероятностью 0,5 соответствующее значение одного из родителей. Если вектора x^1, x^2 совпадали скажем по первой координате, то вектор x' *унаследует* это значение. Геометрически, оператор скрещивания случайным образом выбирает в гиперкубе вершину x', которая принадлежит минимальной грани, содержащей вершины x^1, x^2. Можно сказать, что оператор скрещивания старается выбрать новое решение x' где-то между x^1, x^2, полагаясь на удачу. Более аккуратная процедура могла бы выглядеть таким образом. Новым решением x' является оптимальное решение исходной задачи на соответствующей грани гиперкуба. Конечно, если расстояние Хемминга между x^1, x^2 равно размерности гиперкуба, то задача оптимального скрещивания совпадает с исходной. Тем не менее даже приближенное решение этой задачи вместо случайного выбора заметно улучшает работу генетических алгоритмов [31, 21, 28]. Адаптация этой идеи к двухуровневой задаче о p-медиане будет рассмотрена ниже.

Оператор мутации, применяемый к решению x' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0, 1)$ меняет значение каждой координаты на противоположное. Например, вероятность того, что вектор $(0, 0, 0, 0, 0)$ в ходе мутации перейдет в вектор $(1, 1, 1, 0, 0)$, равна $p_m^3 \times (1 - p_m)^2 > 0$. Таким образом, с ненулевой вероятностью решение x' может перейти в любое другое решение, что гарантирует в асимптотике получение точного решения задачи. Отметим, что модификация решения x' 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 [30]. Это наблюдение известно как тезис о существовании "большой долины" для задач на минимум или "центрального горного массива" для задач на максимум. Этот тезис отчасти объясняет работоспособность генетических алгоритмов. Если в популяции собираются локальные оптимумы, которые сконцентрированы в одном месте, и очередное решение x' 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 области, 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 [51, 52].

3.3.2 Генетический локальный поиск для задачи МПК

Рассмотрим варианты адаптации общей схемы генетических алгоритмов к задаче МПК. В качестве пространства поиска естественно взять p-й слой гиперкуба B^m, т. е. все вектора $y_i \in \{0, 1\}, i \in I$, удовлетворяющие условию $\sum_{i \in I} y_i = p$, а в качестве функции пригодности $\Phi(y)$ — целевую функцию

$F(y)$. Такой выбор функции пригодности представляется обоснованным, т. к. она 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 [43], 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 1. Среди претендентов выбирается та координата, для которой уменьшение целевой функции будет наибольшим. Если уменьшение целевой функции невозможно таким способом, 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 [45]. Согласно правилам этого оператора выбирается одна координата, скажем i_0, и полагается $y_i' = y_i^1$ для $i \leq i_0$ 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 $y^1 = \overline{y}^1, \overline{y}^2, \ldots, \overline{y}^k = y^2$ таких, что расстояние Хемминга между соседними решениями равно 2. 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 [36]. 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 [65]. Если такой выбор невозможен, то в качестве y' берется решение из середины последовательности.

Оператор мутации преобразует случайным образом 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 распределением получает значение 1.

К решению y'' применяется процедура локального спуска сначала по окрестности N_1, а затем по окрестности Лина-Кернигана. 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.

3.3.3 Экспериментальные исследования

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, [51]) и имела низкую плотность: ровно 10 элементов в каждом столбце и строке имели значения от 0 до 4, целые. Остальные элементы полагались равными достаточно большому числу. Матрица приоритетов (g_{ij}) формировалась в два этапа. Сначала каждый столбец матрицы (c_{ij}) упорядочивался по неубыванию $c_{i_1 j} \leq c_{i_2 j} \leq \ldots \leq c_{i_m j}$ 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 выбиралось случайным образом три пары элементов и они менялись друг с другом своими значениями. 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 [77] 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.

3.3.4 Выбор параметров алгоритма

Поведение генетического алгоритма существенно зависит 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 при разных способах селекции и скрещивания. На рис. 3.1 показано изменение этих значений в ходе эволюции при турнирной селекции, вероятность мутации — 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. Для выбора каждого родителя из популяции случайным образом с равномерным распределением выбиралось 3 элемента и лучший из них по целевой функции становился родителем. 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, наверное, объясняется необычайной сложностью самого класса тестовых примеров, где нахождение глобального оптимума или хорошего приближения представляется весьма сложным делом. Жадные стратегии или стратегии связывающих путей, показавшие хорошие результаты на других задачах [36], здесь явно проигрывают.

На рис. 3.2 и 3.3 представлены результаты расчетов для двух других способов селекции. Стратегия "лучший и случайный" проигрывает двум другим, а равновероятная селекция дает, как и равновероятное скрещивание, наилучшие результаты. Интересно отметить, что повышение вероятности мутации (рис. 3.4, 3.5) приводит к сглаживанию различий между операторами скрещивания и даже отсутствие скрещивания (рис. 3.5, 3.6) при наличии высокой вероятности мутации приводит к хорошим результатам. Тем не менее, удаление из алгоритма операторов скрещивания не является оправданным (см. рис. 3.6). Для примеров Резенде и Вернека большой размерности, $n = m = 500$, $p = 50$ [65], генетический алгоритм с любым из рассмотренных операторов скрещивания дает лучшие результаты, чем без этих операторов.

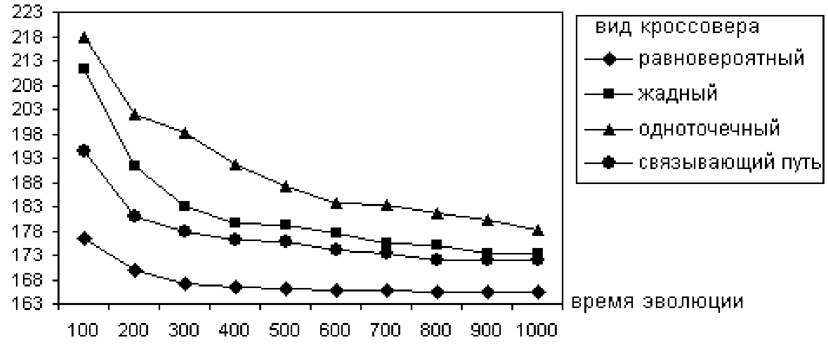

Рис. 3.1: Средние значения F^* при турнирной селекции, $p_m = 0,02$

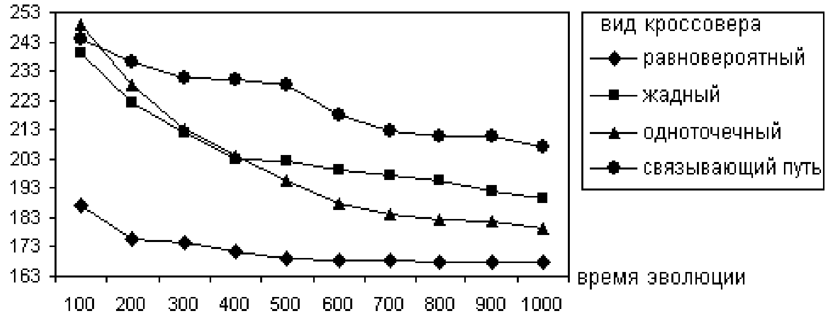

Рис. 3.2: Средние значения F^* при селекции "лучший и случайный", $p_m = 0,02$

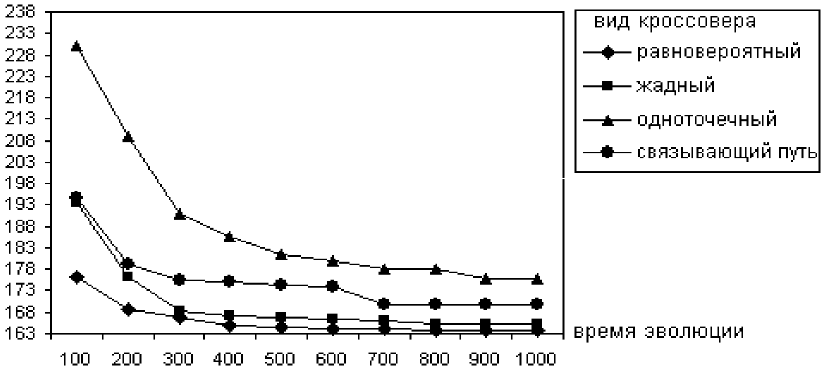

Рис. 3.3: Средние значения F^* при равновероятной селекции, $p_m = 0,02$

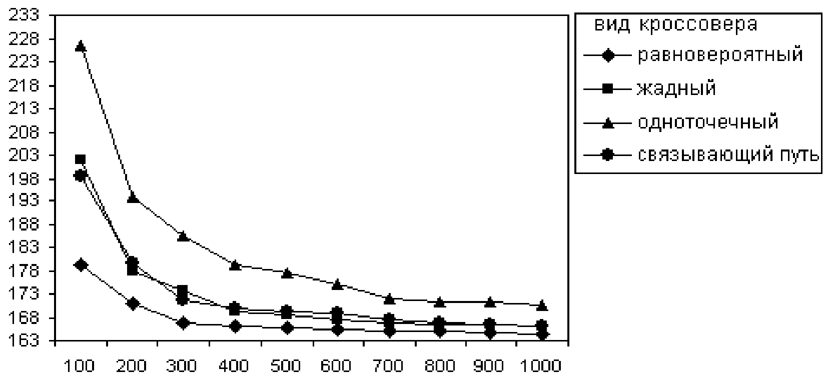

Рис. 3.4: Средние значения F^* при турнирной селекции, $p_m = 0,1$

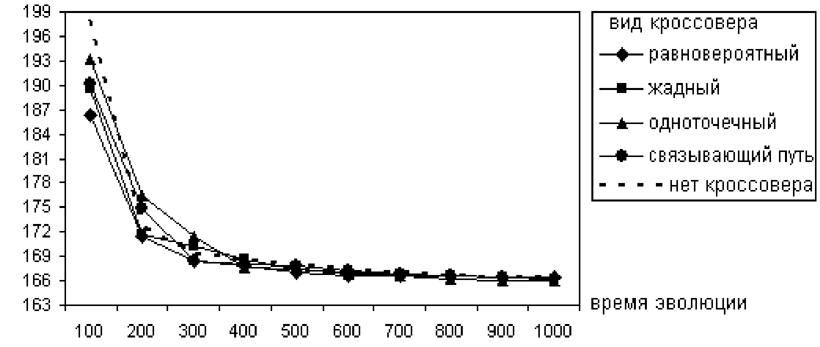

Рис. 3.5: Средние значения F^* при турнирной селекции, $p_m = 0,5$

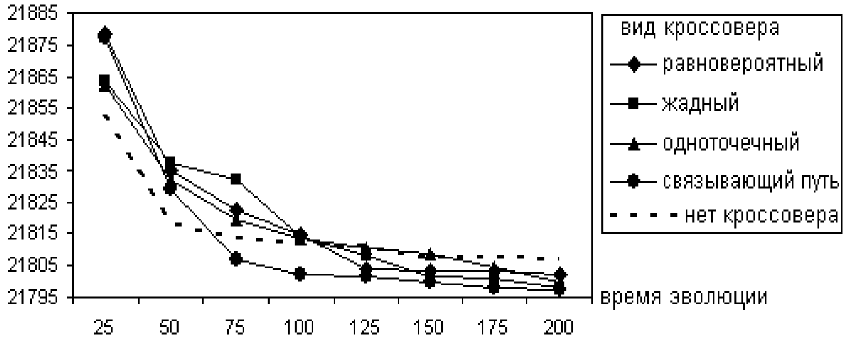

Рис. 3.6: Средние значения F^* при турнирной селекции, $p_m = 0,5$, класс *Euclidean*

3.3.5 Сравнение нижних оценок

В таблице 1 приводятся результаты расчетов для тридцати тестовых примеров из электронной библиотеки. Наряду с указанными нижними оценками линейного программирования приведены еще оценки LB_7, получающиеся 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 разработанным генетическим алгоритмом (GA), 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 коммерческим пакетом за несколько часов работы. Разрыв между оптимумом и нижней оценкой достаточно велик, что оставляет поле для дальнейших исследований, а именно: нахождение другой более сильной нижней оценки.

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

Таблица 3.1: Сравнение верхних и нижних оценок

Глава 4

Задача выбора оптимального парка сельскохозяйственных машин

В данной главе приводится описание системы поддержки принятия решений (СППР), разработанной совместно с группой учёных в ходе выполнения в институте математики им. С.Л. Соболева СО РАН прикладной НИР по заказу одной из крупных компаний, выпускающих машины для сельского хозяйства. Интеллектуальным ядром СППР является математическая модель двухуровневого целочисленного программирования, тесно связанная с задачей МПК. Использование разработанной СППР позволило сократить номенклатуру выпускаемых сельскохозяйственных машин и увеличить прибыль компании.

4.1 Постановка задачи на содержательном уровне

Рассмотрим задачу оптимизации парка сельскохозяйственных машин, выпускаемых крупной машиностроительной компанией. Будем предполагать, что известен список $I = \{1, \ldots, m\}$ типов машин, выпуском которых занимается или могла бы заниматься данная компания, а также множество $J = \{1, \ldots, n\}$ групп клиентов, покупающих эти машины. Под группой клиентов будем понимать покупателей, объединенных общими производственными целями (например, выращивание зерна в черноземной зоне) и характеризующихся одинаковым поведением на рынке сельхозтехники. В частности, покупатели из одной группы предпочитают один и тот же тип машин, если он есть в продаже, или готовы заменить его на другой согласно предпочтениям, одинаковым для всех покупателей из данной группы, если этот тип отсутствует в продаже. Предполагается, что эти предпочтения распространяются не на все типы из списка I, а только на его часть $I_j \subseteq I, j \in J$, т. е. покупатель из группы j не покупает ничего у данной компании, если она не выпускает машины из списка I_j. Другими словами, все покупатели из группы j уйдут к конкурентам компании, если машины из списка I_j выпускаться не будут. Если же покупатель нашел нужную машину в списке выпускаемой продукции, то он делает заказ и тем самым приносит доход компании.

Производство и продажа машин требует от компании определенных затрат, которые состоят из трех частей: затраты на выпуск машин каждого типа, затраты на организацию производства заданного подмножества типов машин и фиксированные затраты на содержание самой компании. Чтобы не потерять клиентов компания должна стремиться выпускать как можно больше типов машин. Однако, в этом случае вторая часть в затратах может оказаться неоправданно большой и прибыль компании упадет. Задача состоит в том, чтобы выбрать такое подмножество $S \subseteq I$ типов машин, чтобы несмотря на потерю части клиентов добиться максимальной прибыли от продаж.

4.2 Предпочтения клиентов

Будем предполагать, что для клиентов каждой группы в списке машин I найдется наиболее подходящая машина, которую они готовы купить для своего бизнеса. Такую машину будем называть "идеалом". В противном случае, они не являются клиентами данной компании и их можно не рассматривать. Выбор наиболее подходящей машины можно осуществить на основании предыстории продаж или на основании аналитических прогнозов о будущих потребностях заказчиков.

Наиболее сложной частью является описание предпочтений клиентов в случае, если "идеал" присутствует в списке I, но не выпускается компанией. Например, некоторый тип машин ранее приобретавшийся клиентами данной группы вдруг исчез из списка продаваемых. Как в этом случае будет вести себя клиент? Чтобы ответить на этот вопрос, рассмотрим составные части машины и факторы, влияющие на выбор клиентов. Известно, что машина состоит из следующих составных частей: двигатели, трансмиссия, ходовая часть и др. Компания выпускает несколько типов двигателей, несколько типов трансмиссий и т. д. Разные типы машин, по сути, получаются разными комбинациями указанных составных частей. Конечно, не все составные части подходят друг к другу, и существуют логические правила, позволяющие ответить на вопрос, можно ли из заданных составных частей собрать машину или это невозможно по конструктивным соображениям. Например, слишком слабый двигатель для большой машины. Тем не менее список гипотетически возможных машин или конфигураций, как их называют конструкторы, чрезвычайно велик и превышает 10^9 наименований для солидной фирмы, выпускающей сельскохозяйственную технику. Итак, предположим, что клиент не нашел в списке продаваемых машин свой "идеал". Тогда он вынужден либо купить другую машину, либо обратиться к другой компании. Как происходит процесс принятия решений?

Будем предполагать, что каждый клиент ищет среди выпускаемых машин наиболее "похожие" на его "идеал" и оценивает их по двум показателям: полезность и стоимость. Припишем каждой составной части значение полезности. Будем считать, что клиент готов рассматривать вариант замены его "идеала" на другую машину, если
1) новая машина может быть получена из "идеальной" заменой отдельных составных частей, и такая замена предусмотрена конструкцией "идеала" машины;
2) сумма квадратов отклонений полезности от "идеальных" значений не превосходит заданного порога;
3) новая машина дороже "идеальной" не более, чем в указанное число раз.

Сформулированные условия позволяют для каждой машины из списка I построить список I_j возможных замен, которые клиент j готов рассматривать, как потенциальный объект для покупки, если данная машина была его "идеалом".

Определим предпочтения клиентов j-ой группы на множестве I_j. Будем считать, что клиент на первое место ставит свою идеальную машину $i(j) \in I$, а все остальные упорядочивает по следующему правилу: сначала элементы множества I_j упорядочиваются по неубыванию суммы квадратов отклонений полезности. Затем, машины с равными значениями этих суммарных отклонений упорядочиваются по неубыванию стоимости.

Последнее, что нужно определить, это спрос при новом предложении. Будем считать, что в каждой группе клиентов имеется определенное число так называемых "консерваторов", которые готовы покупать только свой "идеал" и не согласны ни на какие замены. Если компания закрывает производство "идеала", то они уходят к конкурентам компании и считаются потерянными клиентами. Остальные клиенты данной группы ищут замену своей конфигурации и, если находят подходящую замену из списка I_j в числе продаваемых, то покупают её. В противном случае, если ни одна из возможных замен не выпускается компанией, то они тоже обращаются к конкурентам, увеличивая число потерянных для данной компании клиентов.

4.3 Математическая постановка задачи

Обозначим через $d_i, i \in I$ доход компании от продажи одной машины i-го типа (рыночная стоимость машины). Будем считать, что производственные затраты по выпуску одной машины i-го типа состоят из двух частей:
$c_i > 0$ – прямые затраты на материалы, сдельную оплату труда и прочее;
$c > 0$ – амортизационные отчисления, удельные капитальные затраты и средние удельные затраты на полуфабрикаты, их хранение, доставку и прочее.

Таким образом, прибыль от продажи одной машины i-го типа есть разность $d_i - (c_i + c), i \in I$.

Пусть булевы переменные $y_i \in \{0, 1\}, i \in I$ определяют номенклатуру выпускаемых машин:

$$y_i = \begin{cases} 1, & \text{если машины } i\text{-го типа выпускаются компанией;} \\ 0 & \text{в противном случае.} \end{cases}$$

Булевы переменные $x_{ij} \in \{0, 1\}, i \in I, j \in J$ определяют заказы клиентов на машины:

$$x_{ij} = \begin{cases} 1, & \text{если клиенты } j\text{-ой группы заказывают машины } i\text{-го типа;} \\ 0 & \text{в противном случае.} \end{cases}$$

Определим затраты компании на организацию производства с помощью функции $f(\sum_{i \in I} y_i) \geq 0$. Эта функция равна нулю, если компания не выпускает сельскохозяйственных машин, и монотонно возрастает с ростом аргумента. Как правило, функция $f(\sum_{i \in I} y_i)$ является вогнутой, что соответствует падению удельных затрат на организацию производства с ростом номенклатуры выпускаемой продукции. Однако, в дальнейшем свойства этой функции использоваться не будут.

Обозначим через $\varphi_j \geq 0, \quad j \in J$ число клиентов в j-ой группе, через $g_{ij}, \ i \in I, \quad j \in J$ предпочтения клиентов этой группы на множестве I_j. Будем предполагать, что все элементы g_{ij} различны в каждом столбце этой матрицы, и клиент предпочитает машину i_1 машине i_2, если $g_{i_1 j} > g_{i_2 j}$. Число консервативных клиентов в j-ой группе обозначим через $\psi_j, \quad 0 \leq \psi_j \leq \varphi_j, \quad j \in J$. Напомним, что такие клиенты согласны покупать только "идеальную" конфигурацию $i(j)$ и не согласны на любую другую.

С использованием введенных обозначений задача выбора оптимального парка сельскохозяйственных машин может быть представлена в виде задачи двухуровневого программирования: найти

$$\max_y \{ \sum_{j \in J} \sum_{i \in I_j} (\varphi_j - \psi_j)(d_i - c_i - c) x^*_{ij}(y) + \tag{4.1}$$

$$\sum_{j \in J} \sum_{i \in I_j} \psi_j (d_i - c_i - c) x^*_{i(j)j}(y) - f(\sum_{i \in I} y_i) \}$$

при ограничениях:

$$y_i \in \{0, 1\}, i \in I, \tag{4.2}$$

где $x^*_{ij}(y)$ – оптимальное решение задачи клиентов:

$$\max_x \sum_{j \in J} \sum_{i \in I_j} g_{ij} x_{ij} \tag{4.3}$$

при ограничениях:

$$\sum_{i \in I_j} x_{ij} \leq 1, \quad j \in J, \tag{4.4}$$

$$x_{ij} \leq y_i, \quad i \in I_j, \quad j \in J, \tag{4.5}$$

$$x_{ij} = 0, \quad i \notin I_j, \quad j \in J, \tag{4.6}$$

$$x_{ij} \in \{0, 1\}, \quad i \in I, \quad j \in J. \tag{4.7}$$

Целевая функция задачи верхнего уровня задает суммарную прибыль компании. Целевая функция задачи нижнего уровня отражает предпочтения клиентов. Отличие этой задачи от задачи (1.1)–(1.7) состоит, во-первых, в том, что задачи на верхнем и нижнем уровнях являются задачами максимизации. Во-вторых, в новой задаче отсутствует требование обслужить всех клиентов. В-третьих, заранее не известно число выпускаемых типов машин (отсутствует ограничение 1.2).

4.4 Сведение к задаче МПК

Метод решения задачи (4.1)–(4.7) основан на сведении её к решению серии задач (1.1)–(1.7). Так как функция $f(\sum_{i\in I} y_i)$ может принимать не более m значений, то можно последовательно для каждого значения $p = \sum_{i\in I} y_i$ найти оптимальное решение получающейся подзадачи и затем выбрать из них наилучшее. Таким образом, можно считать, что последнее слагаемое в целевой функции (4.1) является константой, и в задаче верхнего уровня появляется одно дополнительное ограничение $\sum_{i\in I} y_i = p$.

Приведем подобные члены в целевой функции (4.1):

$$\sum_{j\in J}\sum_{i\in I_j}(\varphi_j-\psi_j)(d_i-c_i-c)x^*_{ij}+\sum_{j\in J}\sum_{i\in I_j}\psi_j(d_{i(j)}-c_{i(j)}-c)x^*_{ij}=$$
$$\sum_{j\in J}\sum_{i\in I_j}\{(\varphi_j-\psi_j)(d_i-c_i-c)+\psi_j(d_{i(j)}-c_{i(j)}-c)\}x^*_{ij}$$

и определим матрицу (c_{ij}) следующим образом:

$$c_{ij}=\begin{cases}-\{(\varphi_j-\psi_j)(d_i-c_i-c)+\psi_j(d_{i(j)}-c_{i(j)}-c)\}, & \text{для } j\in J,\ \ i\in I_j;\\ 0, & \text{для } j\in J,\ \ i\notin I_j.\end{cases}$$

Тогда задача нахождения максимума в (4.1) эквивалентна нахождению минимума для новой целевой функции:

$$\sum_{j\in J}\sum_{i\in I}c_{ij}x^*_{ij}(y).$$

Заметим, что теперь вторая сумма включает все элементы множества I, и ограничения (4.4) можно заменить на равенства

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

Меняя знак у матрицы (g_{ij}) и подставляя min в (4.3) вместо max получаем сведение исходной задачи (4.1)–(4.7) к серии задач вида: найти

$$\min_y \sum_{j\in J} \sum_{i\in I} c_{ij} x^*_{ij}(y) \tag{4.8}$$

при ограничениях:

$$\sum_{i\in I} y_i = p \tag{4.9}$$

$$y_i \in \{0,1\}, i \in I, \tag{4.10}$$

где $x^*_{ij}(y)$ – оптимальное решение задачи клиентов:

$$\min_x \sum_{j\in J} \sum_{i\in I} (-g_{ij}) x_{ij} \tag{4.11}$$

$$\sum_{i\in I} x_{ij} = 1, \quad j \in J, \tag{4.12}$$

$$x_{ij} \le y_i, \quad i \in I, \quad j \in J, \tag{4.13}$$

$$x_{ij} \in \{0,1\}, \quad i \in I, \quad j \in J. \tag{4.14}$$

Для решения полученной задачи применяется разработанный в третьей главе алгоритм генетического локального поиска.

4.5 Структура системы поддержки принятия решений

Задача (4.8)–(4.14) является интеллектуальным ядром разработанной СППР по выбору рационального парка выпускаемых сельскохозяйственных машин. СППР является приложением операционной системы Windows 2000 или Windows XP. Она проста и удобна в работе, не требует от пользователя знаний языков программирования и специальной математической подготовки. Работа СППР предполагает использование большого объема исходной информации, включающей описание различных типов конфигураций и правил их конструирования, матрицы прибыли и приоритетов заказчиков по отношению к каждому типу конфигурации и другие показатели. Поэтому для эффективной работы системы желательно наличие не менее 512Mb оперативной памяти и процессор не ниже Pentium 1.2 Ghz.

Рис. 4.1: Подготовка исходных данных

Для удобства вся исходная информация должна быть представлена единообразным способом. Одним из наиболее удобных в этом смысле способов хранения и использования информации являются электронные таблицы. В качестве таких таблиц используются электронные таблицы Excel.

Разработанная СППР состоит из трех основных блоков:

– подготовка исходных данных и проверка их корректности;

– блок оптимизации, предназначенный для проведения многовариантных расчетов по модели (4.8)–(4.14);

– анализ полученных результатов и формирование отчетов для лица, принимающего решение.

Каждому блоку соответствует свое окно с управляющими параметрами и возможностью продолжить расчеты или вернуться к предыдущему шагу.

4.5.1 Подготовка исходных данных

Начиная работу с СППР, пользователь получает доступ к формированию списка клиентов и чтению основных исходных данных: составные части конфигураций, их полезности, коды, стоимости, правила конструирования машин путем замены отдельных составных частей и др. Нажав кнопку "Open Data File" пользователь имеет возможность выбрать нужный вариант исходных данных, а затем нажав кнопку "Read" прочитать эти данные. Так как исходные данные носят разнородный характер и собираются из разных подразделений компании, то в СППР предусмотрен автоматический контроль исходных данных. Панель "Verification messages" показывает пользователю сообщения об обнаруженных ошибках, а нажатие кнопки "View Errors" позволяет детально разобраться в возникающих проблемах при чтении информации. Если ошибок слишком много, то пользователь может остановить процесс чтения информации. Для этого достаточно поставить отметку в позиции "Stop on Error" на панели чтения данных. Как только информация о машинах и их составных частях загружена в оперативную память, появляется возможность чтения данных о клиентах. Для этих целей служит кнопка "Read Customer Demand". Основу этой информации составляют данные о продажах машин за предшествующие периоды, а также статистические данные, позволяющие оценить предпочтения клиентов.

Как только получена вся исходная информация, пользователь может приступать к формированию данных для математической модели. Наиболее трудным этапом этой работы является создание множества $I-$ выпускаемых и гипотетически возможных типов машин. Для этих целей служит кнопка "Create Configurations" и две панели "Features" и "Base and Option Codes". На первой панеле содержится список основных групп унифицированных составных частей, а на второй – список самих составных частей, разбитых на группы. Основная проблема на этом этапе состоит в том, что гипотетически возможных машин слишком много, более 10^9 для реальных исходных данных. Решение задачи (4.8)–(4.14) на таком множестве I практически невозможно на современных компьютерах. Для сокращения размерности в СППР предусмотрена возможность блокировать создание некоторых конфигураций. Для этих целей используются пометки на панелях "Features" и "Base and Option Codes". Снятие пометок в этих окнах запрещает соответствующие замены. Панель "Configurations" показывает результаты формирования множества I: число групп, в которых разрешались замены (Number of Features) и общее число полученных конфигу-

Рис. 4.2: Завершение подготовки исходных данных

раций (Number of Configurations). Если пользователь добился желаемых результатов по сокращению размерности задачи и хочет перейти от этапа формирования данных к этапу оптимизации, он должен нажать кнопку "Optimize", которая первоначально скрыта от пользователя и появляется только по завершению подготовки данных математической модели (см. рис. 4.2).

4.5.2 Блок оптимизации

Для управления поиском рационального парка сельскохозяйственных машин служит окно "Optimization"(рис. 4.3). В правом верхнем углу этого окна пользователю выдается информация об общем числе возможных конфигураций (All Configurations), числе продаваемых ранее конфигураций (Configurations Sold) и числе сегментов (Customer Segments), на которые делится каждая группа клиентов для детального формирования их предпочтений. В каждом сегменте каждой группы предпочтения клиентов одинаковы. В левом верхнем окне "Admissible number of configurations" пользователь задает допустимое число типов конфигураций (величина p

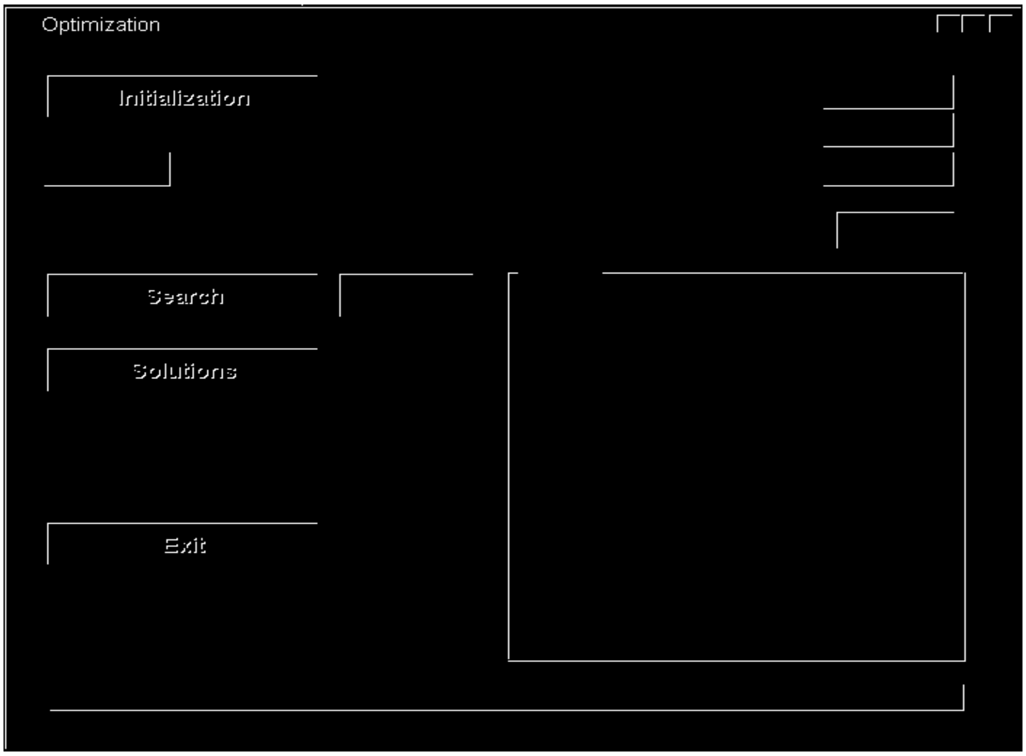

Рис. 4.3: Блок оптимизации

в ограничении 4.9). Работа пользователя начинается с построения матриц (c_{ij}) и (g_{ij}), кнопка "Initialization". Как только данные для математической модели сформированы можно начинать процесс решения задачи. Кнопка "Search" служит для запуска генетического алгоритма, а кнопка "Algorithm" для настройки управляющих параметров. Нажатие кнопки "Algorithm" приводит к появлению нового окна "Optimization Algorithm", на котором расположены следующие управляющие параметры (рис. 4.4):
– длительность эволюции (Evolution Time);
– размер популяции (Population Size);
– способ выбора родительской пары (операторы скрещивания): равномерная селекция, турнирная селекция, селекция по правилу "лучший + случайный"(Parents Selection: Uniform, Tournament, The best and random);
– вероятность выбора родителей (Parents Probability);
– выбор оператора скрещивания: равномерный, жадный (Crossover: Uniform, Greedy);
– вероятность мутации (Mutation Probability);
– глубина окрестности Лина-Кернигана (Depth of LK-neighborhood);
– правила замещения для алгоритмов локального спуска: лучший в окрест-

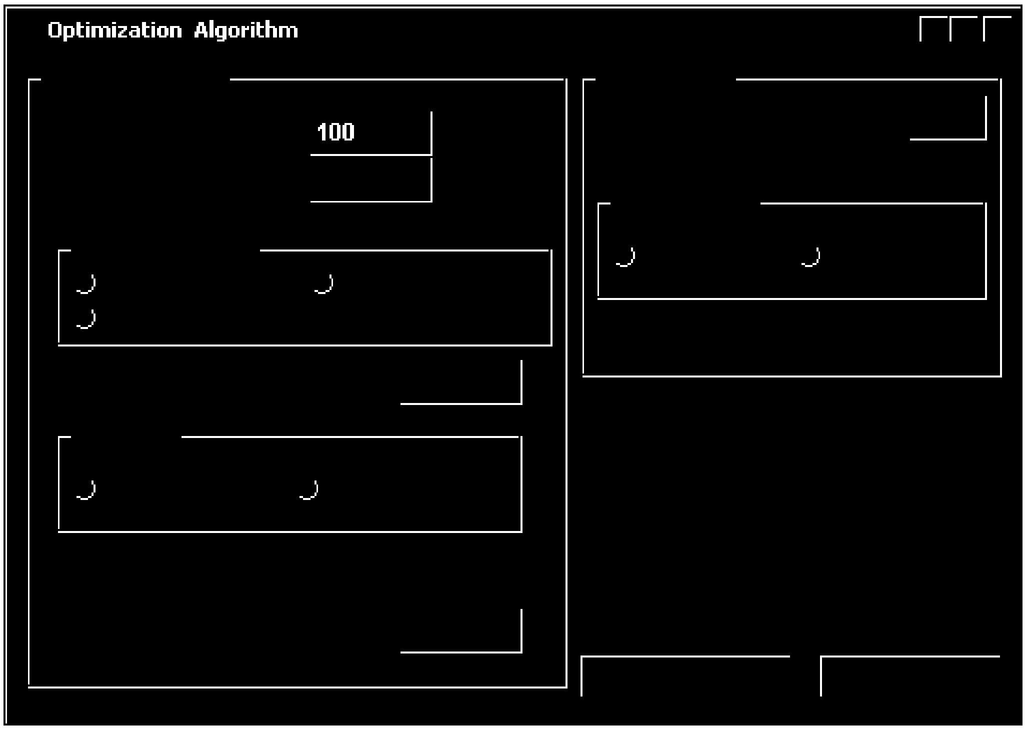

Рис. 4.4: Управляющие параметры алгоритма

ности или первый с улучшением (Pivoting Rules Best, First).

Кнопка "Default" возвращает стандартные установки управляющих параметров алгоритма, кнопка "OK" возвращает пользователя в предыдущее окно "Optimization" и передает алгоритму значения выбранных управляющих параметров.

Нажатие кнопки "Search" приводит к выполнению основных процедур генетического алгоритма. В это время на экране появляется "полоса прогресса" позволяющая оценить, сколько осталось времени до завершения процесса. Если выбранные значения управляющих параметров алгоритма оказались неудачными, и процесс решения идет слишком медленно, пользователь может прервать работу алгоритма, кнопка "Break" и посмотреть на лучшее из найденных к данному моменту решение. Панель "Results" показывает основные характеристики этого решения:

– суммарная прибыль (Total Profit);
– суммарный доход компании (Revenue);
– прямые затраты на производство машин (Total Variable Cost);
– амортизационные отчисления (Total Assert Charge);
– затраты на содержание компании (Overhead);

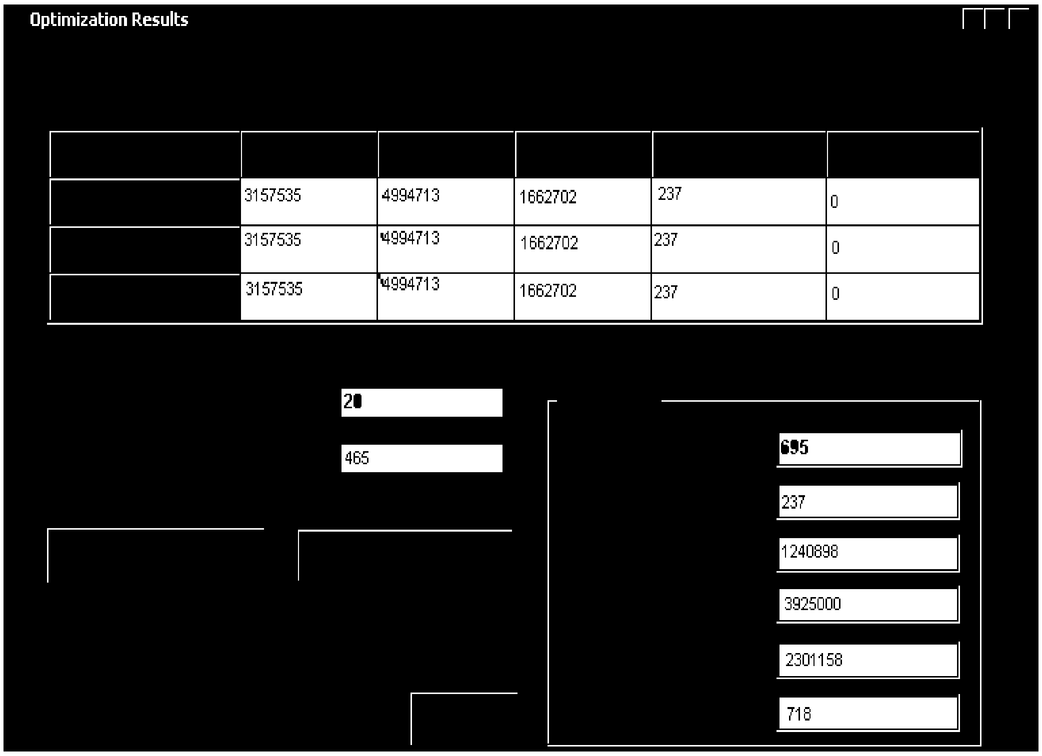

Рис. 4.5: Сравнение полученных решений

– суммарные издержки (Total Expenses);
– число обслуженных клиентов (Customers Served);
– число потерянных клиентов (Customers Lost).

Меняя число допустимых типов конфигураций в ячейке "Admissible Number of Configurations" пользователь может проводить многовариантные расчеты по модели, выбирая наилучшую стратегию по выпуску сельскохозяйственных машин.

4.5.3 Анализ полученных результатов и формирование отчетов

Когда расчеты закончены, наилучший вариант и два ближайших к нему по суммарной прибыли можно получить в развернутом виде, нажав кнопку "Solutions". Эти решения, а также исходный вариант парка машин и их сравнительные характеристики появятся в новом окне "Optimization Results" (рис. 4.5). Подробный список по каждому варианту парка в виде таблиц Excel можно получить, нажав кнопку "Save Solutions and Show". Сравнение различных показателей в найденном решении удобно проводить, используя иллюстрацию решений в виде диаграмм, для этого достаточно

Рис. 4.6: Графическая иллюстрация решений

нажать кнопку "Diagram". В окне "Diagram" имеются встроенные компоненты, каждая из которых позволяет строить различные виды диаграмм (плоскую, трехмерную, в виде графика, графика с областями и другие), сохранять результаты и печатать их. Кнопка "Exit" позволяет пользователю закончить работу с СППР.

Заключение

1. Исследована новая задача о p-медиане с предпочтениями клиентов. Экспериментально установлено, что алгоритмы локального спуска в среднем приводят к решениям с малой относительной погрешностью, хотя в худшем случае такая погрешность может оказаться сколь угодно большой величиной. Исследовано влияние различных правил замещения на число итераций алгоритма локального спуска и погрешность получаемых локальных оптимумов. Предложено новое правило замещения, приводящее к решениям с меньшей погрешностью, чем другие известные правила.

2. Установлено, что целочисленное решение является локальным оптимумом для окрестности N_1 тогда и только тогда, когда оно удовлетворяет условиям Куна-Таккера. Показано, что эти условия равносильны существованию множителей Лагранжа, для которых целочисленное решение исходной задачи является седловой точкой относительно окрестности N_1.

3. Для решения задачи о p-медиане с предпочтениями клиентов разработан генетический алгоритм с апостериорной оценкой точности. Исследованы три формулировки задачи в виде задачи ЦЛП. Показано, что одна из этих формулировок приводит к лучшей нижней оценке, чем две другие.

4. Для задачи оптимизации парка сельскохозяйственных машин разработана система поддержки принятия решений, интеллектуальным ядром которой является математическая модель задачи о p-медиане с предпочтениями клиентов на максимум. Проведенные тестовые расчеты на реальных исходных данных показали высокую эффективность разработанного математического аппарата.

Литература

[1] Береснев В. Л., Гимади Э. Х., Дементьев В. Т. Экстремальные задачи стандартизации. Новосибирск: Наука, 1978.

[2] Береснев В. Л. Дискретные задачи размещения и полиномы от булевых переменных. Новосибирск: Изд-во института математики, 2005.

[3] Береснев В. Л. Об одном классе задач оптимизации параметров однородной технической системы // Управляемые системы. Новосибирск: Институт математики СО АН СССР, 1971. Вып. 9. С. 65–74.

[4] Гермейер Ю. Б. Игры с непротивоположными интересами. Москва: Наука, 1976.

[5] Гимади Э. Х. Эффективный алгоритм решения задачи размещения с областями обслуживания, связанными относительно ациклической сети // Управляемые системы. Новосибирск: Ин-т математики СО АН СССР, 1983. Вып. 23. С. 12–23.

[6] Горбачевская Л. Е. Полиномиально разрешимые и NP-трудные двухуровневые задачи стандиртизации. Кандидатская диссертация физ.-мат. наук, Институт математики им. С.Л.Соболева. Новосибирск, 1998.

[7] Горбачевская Л. Е. Алгоритмы и сложность решения двухуровневых задач стандартизации с коррекцией дохода // Дискрет. анализ и исслед. операций. Сер. 2. 1998. Т. 5, № 2. С. 20–33.

[8] 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.

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

[10] Еремеев А. В. Разработка и анализ генетических и гибридных алгоритмов для решения задач дискретной оптимизации. Кандидатская диссертация физ.- мат. наук. Омск, 2000.

[11] Еремеев А. В. Генетический алгоритм для задачи о покрытии // Дискрет. анализ и исслед. операций. Сер. 2. 2000. Т. 7, № 1. С. 47–60.

[12] Кочетов Ю. А., Обухоская П. А., Пащенко М. Г. Составление расписаний учебных занятий при достаточном числе аудиторий // Труды ИВМиМГ СО РАН. Серия Информатика. Новосибирск 2007. С. 105–112

[13] 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.

[14] Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир. 1985.

[15] Растригин Л. А. Случайный поиск — специфика, этапы истории и предрассудки // Вопросы кибернетики. Вып. 33. 1978. С. 3–16.

[16] Схрейвер А. Теория линейного и целочисленного программирования. М.: Мир, 1991.

[17] Типовая методика оптимизации многомерных параметрических рядов. М.: Изд-во стандартов, 1975.

[18] Типовая методика оптимизации одномерного параметрического (типоразмерного) ряда. М.: Изд-во стандартов, 1976.

[19] Черенин В. П., Хачатуров В. Р. Решение методом последовательных расчетов одного класса задач о размещении производства // Экономико-математические методы. М.: Наука, 1965. С. 279–290.

[20] Aarts E. H. L, Korst J.H.M., van Laarhoven P. J. M. Simulated annealing . Local Search in Combinatorial Optimization. Chichester: Wiley. 1997. P. 91–120.

[21] Aggarwal C. C., Orlin J. B., Tai R. P. Optimized crossover for maximum independent set // Oper. Res. 1997. V. 45, № 2, P. 226–243.

[22] 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.

[23] Alekseeva E.,Fokin M.,Kochetov Yu. and others. Configuration profit tool and configuration optimizer. User's manual. Novosibirsk, Sobolev Institute of Mathematics. 2004.

[24] Anandalingam G., Friesz T. L. Hierarchucal optimization: an introduction // Ann. Oper. Res. 1992. V. 34, № 1. P. 1–11.

[25] Arya V., Garg N., Khandekar R., Meyerson A.,Munaga K.,Pandit V. Local search heuristics for k-median and facility location problems.// SIAM Journal on Computing. 33. 2004. P. 544–562.

[26] 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.

[27] Avella P., Sassano A., Vasil'ev I. Computational study of large-scale p-median problems. // Math. Program. Ser. A 109. 2007. P. 89–114.

[28] 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.

[29] Balinski M. L. Integer programming: methods, uses, computation. Managment Science 12. 1965. P. 253–313.

[30] 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. 101–114.

[31] Borisovsky P., Dolgui A., Eremeev A. Genetic algorithms for supply management problem with lower-bounded demands // Preprints of the

12th IFAC Symposium "Information Control Problems in Manufacturing 2006", Saint-Etienne, France: Elsevier Science, 2006. V. 3. P. 535–540.

[32] 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.

[33] Burke Ed., Kendall G.(Eds.) Search methodologies. Introductory tutorials in optimization and decision support techiques. Springer. 2005.

[34] Dempe S. Foundational of Bilevel Programming. The Netherlands, Dordrecht: Klumer Academic Publishers. 2002.

[35] Fiduccia C. M., Mattheyses R. M. A linear-time heuristic for improving network partitions // Proc. of the 19-th Design Automation Conference, Los Alamitos, CA: IEEE Comput. Soc. Press, 1982. P. 175–181.

[36] Glover F., Laguna M. Tabu search. Boston: Kluwer Acad. Publ., 1997.

[37] Glover F. Tabu search. P.I // ORSA J. Comput. 1989. V. 1. P. 190–206.

[38] Glover F. Tabu search. P.II // ORSA J. Comput. 1990. V. 2. P. 4–32.

[39] Goldberg D. E. Genetic algorithm in search, optimization and machine learning. Reading, MA: Addison-Wesley. 1989.

[40] 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.

[41] Hammer P.L. Plant Location — a pseudo-boolean approach // Israel J. Technology. 1968. V. 6. № 5. P. 330–332.

[42] Hanjoul P., Peeters D. A facility location problem with clients' preference orderings // Regional Science and Urban Economics. 1987. V. 17, Issue 3. P. 451–473.

[43] Hansen P., Mladenović N. An introduction to variable neighborhood search // Meta-heuristics: advances and trends in local search paradigms for optimization. Boston: Kluwer. Acad. Publ., 1998. P. 433–458.

[44] Hansen P., Mladenović N. Developments of variable neighborhood search. Essays and Surveys of Metaheuristics. Boston: Kluwer Acad. Publ., 2002. P. 415–440.

[45] Holland J. H. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press, 1975.

[46] Ibaraki T., Nonobe K., Yagiura M. (Eds.) Metaheuristics: progress as real solvers. Berlin: Springer, 2005.

[47] Johnson D. S., Papadimitriou C. H., Yannakakis M. How easy is local search? // J. of Computer and System Science. 37. 1988. P. 79–100.

[48] Kariv O., Hakimi S. An algoritmic approach to network Location Problems. The p-medians // SIAM Journal of Applied Mathematics. 37. 1979. P. 539–560.

[49] Kernighan B. W., Lin S. An effective heuristic procedure for partitioning graphs // Bell System Tech. J. 1970. V. 49. P. 291–307.

[50] Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by simulated annealing // Science. 1983. V. 220, P. 671–680.

[51] 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.

[52] Kochetov Yu., Ivanenko D. Computationally difficult instances for the uncapacitated facility location problem. In Metaheuristics: progress as real solvers. Edited by Ibaraki T. etc. Berlin: Springer, 2005. P. 351–367.

[53] Korte B., Vygen J. Combinatorial Optimization. Theory and Algorithms. Third Edition. P. 537–571.

[54] Krarup J., Pruzan P. M. The simple plant location problem: survey and synthesis // European J. Oper. Res. 1983. V. 12, № 12. P. 36–81.

[55] Krentel M. W. Structure in locally optimal solutions. // 30th Annual Symposium on Foundation of Computer Science. IEEE Computer Society Press . Los Alamitos CA. P. 216–222.

[56] Krentel M. W. On finding and verifying locally optimal solutions. // SIAM Journal on Computing. 1990. № 19. P. 742–751.

[57] Martelo S., Toth P. Generalized assigment problem. Knapsack problem. Algorithms and Computer Implementations. John Wiley and Sons Ltd. 1990. P. 189–213.

[58] Mirchandani P. B. The p-median problem and generalization. Discrete Location Theory. Edited by Mirchandani P. B, Francis R. L. John Wiley and Sons. 1990. P. 55–119.

[59] Mladenović N.,Brimberg J.,Hansen P., Moreno-Pérez J. The p-median problem: A survey of metaheuristic approaches. // European J. of Oper. Res. (to appear)

[60] Mautor T. Intensification neighborhoods for local search methods. Essays and Surveys in Metaheuristics. Operation Research/Computer Science. Edited by Ribeiro C., Hansen P. Kluwer Acad. Publ. 2001. P. 493–508.

[61] Nemhauser G., Wolsey L. Integer and Combinatorial Optimization. John Wiley and Sons. 1988. P. 402.

[62] Papadimitriou C. H. Theory of comlexity. Addison Wesley, 1994.

[63] Pétrowski D.,Taillard S. Metaheuristics for Hard Optimization. Methods and Case Studies. Springer. 2006.

[64] Rolland E., Schilling D.A., Current J. R. An efficient tabu search procedure for the p-median problem // European J. of Oper. Res. № 96. 1996. P. 329-342

[65] Resende M., Werneck R. A hybrid heuristic for the p-median problem // Journal of heuristics. 10(1). 2004. P. 59–88.

[66] Resende M., Werneck R. On the implementation of a swap-based local search procedure for the p-median problem // Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments (ALENEX'03). Edited by Richard E. Ladner. SIAM. Philadelphia. 2003. P. 119–127.

[67] Ribeiro C., Hansen P.(Eds.) Essay and surveys in metaheuristics. Kluwer Academic Publishers. 2002.

[68] Schäffer A. A., Yannakakis M. Simple local search problems that are hard to solve // SIAM J. Comput. 1991. V. 20, N. 1. P. 56–87.

[69] Schwefel H. P. Numerical optimization of computer models. Chichester: Wiley, 1981.

[70] Stackelberg H. V. The theory of market economy. Oxford: Oxford Univ. Press. 1952.

[71] Tovey C. A. Local improvement on discrete structures // Local search in combinatorial optimization. Chichester: Wiley, 1997. P. 57–90.

[72] Teitz M. B., Bart P. Heuristic methods for estimating the generalized vertex median of a weighted graph // J. of Oper. Res. 16(5). 1968. P. 955–961.

[73] Vicente L. N., Calamai P. H. Bilevel and Multilevel Programming: A bibliography Review // Journal of Global Opt. 1994. V. 5. P. 291–306.

[74] Vredeveld T., Lenstra J. K. On local search for the generalized graph coloring problem // Oper. Res. Letters. 2003. V. 31, N. 4. P. 28–34.

[75] Yannakakis M. Computational Complexity. Local Search in Combinatorial Optimization. Edited by Aarts E. and Lenstra J. K. Chichester: John Wiley and Sons. 1997. P. 19–55.

[76] http://www.research.att.com/ mgcr/data/index.html

[77] http://www.gams.com/