РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ

Омский филиал Федерального государственного бюджетного учреждения науки Института математики им. С.Л. Соболева Сибирского отделения Российской академии наук

На правах рукописи

ЕРЕМЕЕВ Антон Валентинович

ИССЛЕДОВАНИЕ ЭВОЛЮЦИОННЫХ МЕТОДОВ РЕШЕНИЯ ЗАДАЧ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ

05.13.17 — теоретические основы информатики

Диссертация на соискание ученой степени доктора физико–математических наук

Научный консультант
д.ф.-м.н., профессор
Колоколов А. А.

Омск 2013

Оглавление

Введение **4**

1 Постановки задач и схемы эволюционных алгоритмов **40**

1.1 Задачи комбинаторной оптимизации 40

1.2 Эволюционные алгоритмы 47

2 Исследование эволюционных алгоритмов с позиций локальной оптимизации **65**

2.1 Генетический алгоритм как метод локального поиска 65

2.2 Динамика численности особей с высокой приспособленностью в популяции генетического алгоритма 74

2.3 Сравнение эволюционных алгоритмов с эволюционной стратегией (1+1)-ES 95

2.4 Статистические оценки числа локальных оптимумов 104

3 Исследование сложности задачи оптимальной рекомбинации **114**

3.1 Постановка задачи 114

3.2 Полиномиально разрешимые случаи 118

3.3 NP-трудные случаи 132

4 Построение эволюционных алгоритмов со свойствами динамического программирования **156**

4.1 Формализация метода динамического программирования . . 157

4.2 Эволюционные алгоритмы на основе динамического программирования 169

4.3 Вполне полиномиальная рандомизированная аппроксимационная схема 176

5 Применение оптимальной рекомбинации в генетических алгоритмах 188

5.1 Задача о наименьшем покрытии 188

5.2 Задача управления поставками продукции 216

5.3 Задача балансировки автоматизированной производственной линии 244

Заключение 265

Литература 267

Введение

Актуальность темы. Задачи комбинаторной оптимизации находят широкое применение в информатике, технике, экономике, планировании и других областях. В настоящее время в комбинаторной оптимизации интенсивно развивается подход, основанный на бионическом принципе моделирования эволюционных процессов адаптации в живой природе. Эволюционные методы успешно применяются в информационных технологиях проектирования, планирования, управления, распознавания образов и т.д. Этим методам посвящено большое число работ как отечественных авторов (Д.И. Батищев, И.Л. Букатова, Н.Г. Загоруйко, А.Г. Ивахненко, Ю.А. Кочетов, Г.С. Лбов, В.М. Курейчик, И.П. Норенков, Ю.И. Неймарк, Л.А. Растригин, В.Г. Редько, Е.С. Семенкин и др.) [7, 8, 18, 55, 74, 76, 81, 84, 88, 89, 95], так и зарубежных (Э. Балаш, Д. Голдберг, Дж. Холланд, И. Реченберг, М. Шоно, П. Витани, М. Воз, И. Вегенер) [122, 132, 197, 213, 266, 288, 289].

Несмотря на большое количество результатов, полученных в области комбинаторной оптимизации, потребность в дальнейших исследованиях не уменьшается. Это связано как с постоянным притоком новых задач, так и со сложностью их решения. Многие задачи комбинаторной оптимизации являются NP-трудными, и построение оптимального решения требует значительных временных затрат даже при сравнительно низких размерностях исходных данных. В таких ситуациях возникает необходимость в более глубоком анализе задач с позиций теории вычислительной сложности, позволяющей оценить перспективы разработки алгоритмов с заданными характеристиками (время вычислений, требуемый объем памяти, точность и др.) и в итоге найти подход к решению задачи.

Важное место в комбинаторной оптимизации занимает проблемати-

ка целочисленного программирования, которая включает вопросы, связанные с теорией двойственности, полиэдральным подходом, методами отсечения, ветвей и границ, декомпозиции, множителей Лагранжа, выпуклым и невыпуклым программированием, устойчивостью решений и т.д. Исследованию данной проблематики посвящены работы Дж. Бендерса, В.Л. Береснева, В.П. Булатова, Р. Гомори, Дж. Данцига, В.Т. Дементьева, Ю.Г. Евтушенко, В.А. Емеличева, И.И. Еремина, М.М. Ковалева, А.А. Колоколова, В.К. Леонтьева, Вл.Д. Мазурова, Дж. Немхаузера, И.В. Сергиенко, А.С. Стрекаловского, Д. Фалкерсона, В.Н. Шевченко, Дж. Эдмондса и многих других авторов как в России, так и за рубежом [11, 12, 19, 32, 33, 45, 78, 97, 103, 106, 187]. В ряде известных методов решения задач целочисленного программирования используется сведение исходной задачи к последовательности задач непрерывной оптимизации. На этом подходе основаны методы отсечения, декомпозиции, ветвей и границ, алгоритмы перебора L-классов и др. [49, 63–66, 93]. Актуальной проблемой при исследовании таких методов является построение нижних и верхних оценок числа итераций, в том числе оценок в среднем. Важные результаты в этом направлении получены Е. Вентой, В.П. Гришухиным, Р. Джерослоу, О.А. Заблоцкой, Л.А. Заозерской, А.А. Колоколовым, Н.Н. Кузюриным, Х. Ленстрой, Ю.Ю. Финкельштейном [26, 28, 48, 53, 67, 69, 75, 221, 235, 252].

При решении задач комбинаторной оптимизации широко используется метод динамического программирования, предложенный Р. Беллманом [130], и его обобщения. Актуалные вопросы распараллеливания, преодоления больших затрат памяти, разработки гибридных алгоритмов, анализа графовых интерпретаций динамического программирования и решения многокритериальных задач в рамках данного подхода рассматриваются в работах У. Бертеле, Ф. Бриоши, В.В. Быковой, З. Галил, Г.Г. Забудского, А.В. Кельманова, Д.И. Когана, Н.Н. Моисеева, В.В. Серваха, Н.З. Шора, О.А. Щербины [20, 49, 57, 61, 79, 80, 104, 131, 142, 188] и других

авторов.

Среди первых методов решения задач комбинаторной оптимизации наряду с методами отсечений, динамического программирования и ветвей и границ возникли методы локального поиска и локальные алгоритмы [47, 87, 204, 236]. В нашей стране основоположниками этого направления стали Ю.И. Журавлев, который ввел в рассмотрение класс локальных алгоритмов и провел анализ их сложности [46], и Л.А. Растригин, предложивший и исследовавший рандомизированные алгоритмы локального поиска [87], а также И.В. Сергиенко, развивший метод вектора спада и обосновавший его работоспособность [93]. Значительные результаты в исследовании возможностей методов локального поиска получены А.Н. Антамошкиным, Б. Керниганом, Ю.А. Кочетовым, С. Лином, Х. Пападимитриу, О.Э. Семенкиной, С. Товеем, М. Яннакакисом и другими [72, 86, 96, 113, 275, 296].

Приближенные алгоритмы решения задач комбинаторной оптимизации оказываются незаменимыми в ситуациях, когда получение точного решения требует чрезмерных временных затрат. Значительный вклад в область разработки и анализа приближенных алгоритмов внесли А.А. Агеев, С. Арора, Э.Х. Гимади, Н.И. Глебов, Д. Джонсон, А.А. Корбут, Г. Корнужолс, Б. Корте, Л. Ловас, В.А. Перепелица, В.К. Попков, П. Рагхаван, И.Х. Сигал, К. Томпсон, М.Ю. Хачай, Д. Хочбаум [2, 23, 27, 31, 82, 92, 102, 115, 212, 216, 230] и ряд других авторов. В настоящее время интенсивно исследуются вопросы существования полиномиальных аппроксимационных схем и их трудоемкости. Существенные результаты в этом направлении получили Г. Воегингер, Г.В. Генс, О. Ибарра, М.Я. Ковалев, В. Кубик, Е.В. Левнер, С.В. Севастьянов, Я.М. Шафранский [21, 22, 60, 215, 233, 293] и др.

Эволюционные алгоритмы (ЭА) берут начало в работах Л. Фогеля, А. Оуэнса и М. Уолша [101] и Дж. Холланда [213], где было предложено моделировать процесс биологической эволюции с целью синтеза эффектив-

ных в некотором смысле структур и создания систем искусственного интеллекта. В нашей стране А.Г. Ивахненко и Л.А. Растригиным независимо были предложены методы случайного поиска, где также использовались идеи эволюции [55, 265]. Характерной особенностью ЭА является имитация процесса эволюционной адаптации биологической популяции к условиям окружающей среды, при этом *особи* соответствуют пробным точкам в пространстве решений задачи оптимизации, а *приспособленность* особей определяется значениями целевой функции и штрафами за нарушение ограничений задачи, если такие имеются. В рамках данного подхода предложены эволюционные стратегии [266], генетические алгоритмы [213], алгоритмы случайного поиска с адаптацией [76], эволюционного моделирования [18,101]), генетического программирования [231], многокритериальные ЭА [257]. К классу ЭА также могут быть отнесены алгоритмы Метрополиса [248], имитации отжига [227], поиска с запретами [194] и др. Указанные алгоритмы различаются способами моделирования эволюционного процесса и отражаемыми аспектами, однако имеют много общих элементов. Области применения этих алгоритмов также несколько различаются.

Принципы наследственности, изменчивости и отбора в эволюционных алгоритмах реализуются при построении новых решений-потомков посредством рандомизированных процедур (операторов), модифицирующих полученные ранее пробные точки подобно процессам мутации и кроссинговера в живой природе. Отбор таких пробных точек производится с учетом значений функции приспособленности (особям, имеющим преимущество по приспособленности, даются бóльшие шансы на отбор в качестве родительских решений). Эволюционные алгоритмы могут применяться как для задач с одним критерием оптимизации, так и для многокритериальных задач, поэтому допускается и многокритериальная вектор-функция приспособленности. Ввиду простоты адаптации вычислительных схем эволюционных алгоритмов эти методы активно применяются для решения задач

комбинаторной оптимизации, возникающих в управлении, планировании, проектировании, распознавании образов и других областях (см., например, [84, 90, 161, 163]).

О применимости ЭА к индивидуальной задаче комбинаторной оптимизации естественно судить по математическому ожиданию времени первого достижения оптимального решения или достаточно точного приближенного решения. Эти средние значения могут быть оценены снизу и сверху функциями от длины записи исходных данных задачи, что позволяет делать вывод об эффективности ЭА на задаче комбинаторной оптимизации в целом. Необходимо отметить, что существование ЭА, обнаруживающего оптимальное решение какой-либо NP-трудной задачи в среднем за время, полиномиально ограниченное сверху, представляется маловероятным (это противоречило бы предположению о неравенстве классов $\mathrm{NP} \neq \mathrm{RP}$, которое длительное время принимается в теории сложности в качестве рабочей гипотезы [229]).

Теоретически наиболее детально исследованы эволюционные алгоритмы с достаточно простыми (как правило, линейными по трудоемкости) операторами мутации и кроссинговера [248]. Как выяснилось, базовые схемы эволюционных алгоритмов при подходящем способе представления решений и настройках параметров могут воспроизводить основные этапы работы известных алгоритмов дискретной оптимизации. Значимые результаты в данном направлении получены при анализе ЭА для таких классических задач дискретной оптимизации, как задача о кратчайшем пути в графе [279], задача об остовном дереве минимального веса [249] и задача о разрезе минимального веса [247]. Оптимальные решения этих задач могут быть получены в среднем за полиномиальное время при помощи многокритериальных эволюционных алгоритмов, воспроизводящих работу алгоритмов Дейкстры и Краскала, или неявно учитывающих двойственность задач о минимальном разрезе и максимальном потоке. ЭА с аналогичны-

ми свойствами найдены и для некоторых других задач [157, 186, 193, 248]. Вместе с тем недостает обобщающих результатов об эффективности ЭА на крупных классах задач, а также теоретически обоснованных выводов о преимуществе одного ЭА над другим на некотором классе задач.

На практике попытки применения стандартных ЭА без учета специфики решаемых задач достаточно быстро показали их низкую эффективность в сравнении со специализированными алгоритмами. Тем не менее гибкость вычислительных схем ЭА и возможности их комбинирования с другими методами позволили разработать конкурентоспособные *гибридные* алгоритмы. В таких алгоритмах отдельные операторы представляют собой уже известные алгоритмы, такие как локальный поиск, жадный алгоритм, метод ветвей и границ или перебор L-классов (см., например, [74, 84, 163, 177]). Для некоторых гибридных ЭА удается и теоретически гарантировать качество получаемого решения благодаря свойствам используемых в них классических методов [163, 177]. К сожалению, в большинстве случаев обоснование работоспособности гибридных ЭА ограничивается применением теоремы Е. Аартс, А. Айбен и К. ван Хи о сходимости к оптимуму «почти наверное» [277], предъявляющей достаточно слабые требования к операторам ЭА, но не дающей оценок сверху на время достижения оптимума. Для выхода из создавшейся ситуации требуется получение оценок времени отыскания решений требуемой точности и вероятности порождения достаточно точных решений на заданной итерации ЭА.

Работоспособность генетического алгоритма (ГА) существенно зависит от выбора оператора кроссинговера, где комбинируются элементы родительских решений. Новым направлением исследований является анализ сложности и разработка эффективных алгоритмов решения *задачи оптимальной рекомбинации* (ЗОР), состоящей в отыскании наилучшего возможного результата кроссинговера при заданных родительских генотипах. Результаты Ч. Аггарвала, Э. Балаша, В. Кука, Дж. Орлина, П. Сеймура

и др. [42, 108, 122, 147, 160, 195] дают экспериментальное подтверждение целесообразности решения ЗОР (точного или приближенного) в операторах кроссинговера. Тем не менее до сих пор не было проведено систематического анализа вычислительной сложности ЗОР и эффекта от ее решения в процессе работы ЭА.

Цель диссертации – развитие и исследование эволюционных методов решения задач комбинаторной оптимизации. Поставленная цель определила следующие основные задачи исследования:

1) разработка эволюционных алгоритмов и их гибридных вариантов с использованием классических методов комбинаторной оптимизации;

2) оценка точности решений, получаемых эволюционными алгоритмами, построение оценок трудоемкости этих алгоритмов и их основных операторов, сопоставление полученных оценок с известными аналогами;

3) анализ сложности задач комбинаторной оптимизации, и в частности, задач оптимальной рекомбинации, оценивание параметров задач, влияющих на работоспособность ЭА.

Методы исследования. При выполнении работы использовались методы математического программирования, теории вероятностей и математической статистики, теории вычислительной сложности, бионические методы и модели, а также современная методология экспериментальных исследований с применением вычислительной техники.

Научная новизна. Введена в рассмотрение общая постановка задачи оптимальной рекомбинации в терминах задач комбинаторной оптимизации; предложен метод построения сводимостей ЗОР, подобных известным сводимостям задач комбинаторной оптимизации, позволивший доказать полиномиальную разрешимость или NP-трудность ЗОР для различных задач. Разработана методика использования в ГА операторов кроссинговера, основанных на решении ЗОР, что дало возможность улучшить известные ранее экспериментальные результаты для ряда задач.

Впервые введено отношение доминирования для операторов мутации и кроссинговера, позволившее при некоторых условиях выделить наилучшего представителя в классе ЭА.

Новым является предложенный подход к получению верхних оценок среднего числа итераций ЭА до нахождения оптимального или приближенного решения через сопоставление этапов развития популяции ЭА с этапами работы алгоритмов локального поиска и динамического программирования.

Благодаря группировке состояний в модели ГА впервые получены оценки числа особей заданного качества в популяции ГА с турнирной селекцией, применимые для ряда задач комбинаторной оптимизации.

Впервые предложены статистические методы построения нижних оценок числа локальных оптимумов задачи комбинаторной оптимизации.

На защиту выносятся следующие основные результаты.

1. Проведено комплексное исследование эволюционных методов с позиций локальной оптимизации: найдены необходимые и достаточные условия, при которых рандомизированный вариант локального поиска является наилучшим методом по вероятности отыскания оптимума в классе эволюционных алгоритмов с заданным оператором мутации; найдены достаточные условия, при которых генетический алгоритм с турнирной селекцией достигает локальный оптимум в среднем за полиномиально ограниченное время, и показано выполнение этих условий на классе задач GLO; предложены статистические методы нахождения нижних оценок числа локальных оптимумов задачи.

2. Исследована сложность задач оптимальной рекомбинации: установлена полиномиальная разрешимость задач оптимальной рекомбинации в генетических алгоритмах для ряда задач булевого линейного программирования, в частности задач упаковки и разбиения множества и простейшей задачи размещения производства; установлена NP-трудность оптимальной

рекомбинации для задачи коммивояжера, задач о рюкзаке, кратчайшем гамильтоновом пути, упаковке в контейнеры и некоторых других классических задач комбинаторной оптимизации.

3. Построены эволюционные алгоритмы, аналогичные по свойствам алгоритмам динамического программирования: получены полиномиальные оценки средней трудоемкости многокритериального эволюционного алгоритма через трудоемкость алгоритма динамического программирования; для класса задач, удовлетворяющих условиям существования вполне полиномиальной аппроксимационной схемы Г. Воегингера, предложена вполне полиномиальная рандомизированная аппроксимационная схема, основанная на эволюционном алгоритме.

4. Разработан подход к построению гибридных генетических алгоритмов, использующих оптимальную рекомбинацию и жадные алгоритмы для решения NP-трудных задач комбинаторной оптимизации: предложены новые генетические алгоритмы для задач покрытия множества, управления поставками продукции и балансировки производственной линии, экспериментально установлены классы задач, на которых такие алгоритмы имеют преимущество по сравнению с известными алгоритмами; выполнен теоретический анализ сложности приближенного решения задачи управления поставками продукции.

5. Проведено исследование генетического алгоритма с турнирной селекцией и мутацией: предложена математическая модель эволюционного процесса в популяции генетического алгоритма; с использованием данной модели построены оценки динамики численности особей с высокой приспособленностью и изучены частные случаи достижимости оценок.

Таким образом, в данной работе проведено комплексное исследование эволюционных методов решения задач комбинаторной оптимизации: выявлены широкие классы задач, для которых существуют эволюционные алгоритмы, аналогичные по свойствам методам локального поиска и дина-

мического программирования; проведен теоретический анализ сложности задач оптимальной рекомбинации и экспериментально показана целесообразность применения оптимальной рекомбинации в гибридных эволюционных алгоритмах; построены оценки вероятностей получения решений заданного качества.

Практическая и теоретическая ценность. Работа имеет теоретический и экспериментальный характер. Полученные в ней результаты позволяют оценить возможноти эволюционных методов и области их применимости. Разработанные алгоритмы могут использоваться в информационных технологиях проектирования и управления производственными и транспортными системами, а также в научных исследованиях для оценки параметров комбинаторных объектов. Материалы диссертации применяются при подготовке студентов и аспирантов.

Апробация работы. Основные результаты и положения работы докладывались автором на следующих научных конференциях:

Всероссийской конференции «Проблемы оптимизации и экономические приложения» (Омск, 1997, 2003, 2006, 2009, 2012);

Всероссийской конференции «Математическое программирование и приложения» (Екатеринбург, 1997, 1999, 2007, 2011);

Российской конференции «Дискретный анализ и исследование операций» (Новосибирск, 1998, 2000, 2002);

Российской конференции «Дискретная оптимизация и исследование операций» (Владивосток, 2007; Алтай, 2010; Новосибирск, 2013);

International Conference on Operations Research (Цюрих, Швейцария, 1998);

Байкальской международной школе-семинаре «Методы оптимизации и их приложения» (Иркутск, 1998, 2001; Северобайкальск, 2008);

Conference Evolution Artificielle (Дункерк, Франция, 1999);

Международном семинаре «Методы дискретной оптимизации»

(Минск, 2000; Омск - Иркутск, 2004);

European Workshop on Evolutionary Computation in Combinatorial Optimization (Милан, Италия, 2001; Турин, Италия, 2011);

International Seminar «Theory of Evolutionary Algorithms» (Дагштул, Германия, 2002, 2004, 2006, 2008, 2010, 2013);

Workshop on the Foundations of Genetic Algorithms (Торремолинос, Испания, 2003);

Conference of the European Chapter on Combinatorial Optimization (Минск, 2005);

Азиатской школе-семинаре «Оптимизация сложных систем» (Новосибирск, 2006; Усть-Каменогорск, 2010);

Международной научной конференции «Дискретная математика, алгебра и их приложения» (Минск, 2009);

Международной конференции «Интеллектуализация обработки информации» (Будва, Черногория, 2012);

Euro Mini Conference on Variable Neighbourhood Search (Херцег Нови, Черногория, 2012);

а также на семинарах в Институте математики им. С.Л. Соболева СО РАН и его Омском филиале, Омском государственном университете им. Ф.М. Достоевского и Вычислительном центре РАН им. А.А. Дородницына. Материалы диссертации легли в основу курса «Эволюционные алгоритмы», читаемого магистрантам Института математики и информационных технологий ОмГУ им. Ф.М. Достоевского.

Публикации. По теме диссертации автором опубликовано 38 статей и глав монографий, в том числе 20 статей в журналах из списка ВАК.

Личный вклад. Диссертационная работа представляет собой единый цикл исследований автора, объединенных не только постановками задач и схемами алгоритмов, но и методами исследований. В совместных работах соискателю принадлежат основные идеи построения новых эволюци-

онных алгоритмов, методов обоснования оценок точности и трудоемкости алгоритмов, а также исследования вычислительной сложности рассматриваемых задач. Отдельные элементы доказательств утверждений и теорем выполнены в соавторстве при непосредственном участии соискателя. Конфликт интересов с соавторами отсутствует.

Объем и структура диссертации. Диссертация состоит из введения, пяти глав, заключения и списка литературы (296 наименований). Объем диссертации 300 страниц.

Содержание работы

Глава 1 содержит постановки задач комбинаторной оптимизации и общие схемы эволюционных алгоритмов, которые многократно используются далее в различных разделах диссертации.

В разделе 1.1 приводится общая постановка задачи комбинаторной оптимизации. Пусть $\mathbb{R}$ и $\mathbb{N}$ – множества вещественных и натуральных чисел соответственно; $\{0,1\}^*$ – множество всевозможных строк из нулей и единиц. Для строки $S \in \{0,1\}^*$ символом $|S|$ обозначается ее длина.

Определение 1.1. *Задача комбинаторной оптимизации – это тройка* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f_I)$, *где*

1) $\mathrm{Inst} \subseteq \{0,1\}^*$ – *множество индивидуальных задач из* Π;

2) $\mathrm{Sol}(I) \subseteq \{0,1\}^{n(I)}$ – *множество допустимых решений индивидуальной задачи* $I \in \mathrm{Inst}$, *где* $n(I)$ – *размерность пространства решений;*

3) *для каждой* $I \in \mathrm{Inst}$ *определена функция* $f_I : \mathrm{Sol}(I) \to \mathbb{R}$, *которую требуется максимизировать (если* Π – *задача максимизации) или минимизировать (если* Π – *задача минимизации).*

Далее через f_I^* обозначается оптимальное значение целевой функции в индивидуальной задаче I. По умолчанию под *полиномиально ограничен-*

ной величиной подразумевается величина, ограниченная сверху полиномом с положительными коэффициентами относительно $|I|$. Задача комбинаторной оптимизации называется полиномиально ограниченной, если значения $f_I(\mathbf{x})$ полиномиально ограничены.

Наибольший теоретический интерес представляют задачи комбинаторной оптимизации из класса NPO, где вводятся следующие «технические» предположения.

Определение 1.2. *Задача комбинаторной оптимизации принадлежит классу* NPO, *если отношения* $I \in \mathrm{Inst}$ *и* $\mathbf{x} \in \mathrm{Sol}(I)$ *могут быть проверены за полиномиально ограниченное время, размерность* $n(I)$ *полиномиально ограничена, а функция* $f_I : \mathrm{Sol}(I) \to \mathbb{N}$ *вычислима за полиномиально ограниченное время для любой* $I \in \mathrm{Inst}$.

Далее в разделе 1.1 рассматривается несколько классических задач комбинаторной оптимизации в качестве примеров: задача об одномерном булевом рюкзаке, задача коммивояжера, задача о наименьшем покрытии, задача о вершинном покрытии и простейшая задача размещения.

Здесь также вводятся стандартные определения, формализующие понятие приближенного решения задачи и вполне полиномиальной аппроксимационной схемы. В завершение раздела излагается общая схема алгоритма локального поиска и связанные с нею определения.

Пусть для всякого $\mathbf{y} \in \mathrm{Sol}(I)$ определена окрестность $\mathcal{N}_I(\mathbf{y}) \subseteq \mathrm{Sol}(I)$. Совокупность $\{\mathcal{N}_I(\mathbf{y}) : \mathbf{y} \in \mathrm{Sol}(I)\}$ называется *системой окрестностей*. Если для $\mathbf{x} \in \mathrm{Sol}(I)$ при всяком $\mathbf{y} \in \mathcal{N}_I(\mathbf{x})$ выполняется неравенство $f_I(\mathbf{y}) \leq f_I(\mathbf{x})$ в случае задачи максимизации, или $f_I(\mathbf{y}) \geq f_I(\mathbf{x})$ в случае задачи минимизации, то $\mathbf{x}$ называется локальным оптимумом. Система окрестностей $\{\mathcal{N}(\mathbf{x}) \mid \mathbf{x} \in \mathrm{Sol}(I)\}$ называется k-ограниченной, если для любых $\mathbf{x} \in \mathrm{Sol}$ и $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ выполнено $\delta(\mathbf{x}, \mathbf{y}) \leq k$, где $\delta(\cdot, \cdot)$ – метрика Хэмминга.

В разделе 1.2 описываются класс эволюционных алгоритмов и наиболее известные его представители.

Первым излагается классический генетический алгоритм (КГА), предложенный Дж. Холландом в [213], как алгоритм, имитирующий процесс адаптации популяции к окружающей среде, взаимодействие с которой задается *функцией приспособленности* особей. На каждой итерации КГА с помощью рандомизированных операторов *мутации* и *кроссинговера* строится новая популяция (поколение). Численность популяции N фиксирована от начала работы алгоритма до конца. В КГА каждая особь текущей популяции выбирается в качестве родительской для построения новой особи-потомка с вероятностью, пропорциональной приспособленности родительской особи с помощью *оператора пропорциональной селекции*.

Процесс работы ГА представляет собой последовательную смену популяций, состоящих из фиксированного числа особей, соответствующих пробным точкам пространства решений. Особи, отвечающие решениям с бóльшим качеством (более приспособленные), получают преимущество в порождении потомков в следующей популяции.

С 80-х годов XX века генетические алгоритмы стали активно использоваться как метод решения задач оптимизации, при этом особи представляют пробные решения задачи, а функция приспособленности в некотором смысле характеризует их «качество». Синонимом термина «особь» является «генотип». Как правило, первый термин используется, когда речь идет об особи в составе популяции, а последний – когда популяция не учитывается.

Генотип ξ представляет собой строку $(\xi_1, \xi_2, \ldots, \xi_l)$ фиксированной длины l. Элементы этой строки – символы некоторого конечного алфавита, их принято называть *генами* по аналогии с генами живых организмов, которые представляют собой участки молекулы ДНК.

Обозначим через $\mathcal{B}$ множество всевозможных генотипов длины l с

символами из заданного набора алфавитов $A_1, \ldots, A_l$, т. е. $\mathcal{B} = A_1 \times \ldots \times A_l$. Для применения ГА к задаче комбинаторной оптимизации необходимо определить *функцию представления решений* $\Psi : \mathcal{B} \to \{0,1\}^n$, задающую способ кодирования элементов пространства решений с помощью генотипов.

Образ $\mathbf{x} = \Psi(\xi)$ принято называть *фенотипом*, соответствующим генотипу ξ. Во многих случаях в ГА используется тот же способ кодировки решений, что и в постановке задачи комбинаторной оптимизации. В таком случае термины «генотип», «фенотип» и «пробное решение» оказываются эквивалентными, а отображение $\Psi(\cdot)$ становится тождественным.

Популяция представляет собой вектор $X = (\xi^1, \xi^2, \ldots, \xi^N) \in \mathcal{B}^N$, элементами которого являются генотипы. Способ нумерации особей в популяции не имеет значения. Популяция поколения t, $t = 0, 1, \ldots$, обозначается через $X^t = (\xi^{1t}, \xi^{2t}, \ldots, \xi^{Nt})$. Итерацией ГА является переход от текущей популяции X^t к следующей популяции X^{t+1}.

В соответствии с общепринятым подходом будем предполагать, что при $\Psi(\xi) \in \mathrm{Sol}$ функция приспособленности имеет вид $\Phi(\xi) = \phi_{\mathrm{fit}}(f(\Psi(\xi)))$, где $\phi_{\mathrm{fit}} : \mathbb{R} \to \mathbb{R}$ – некоторая строго возрастающая функция, если решается задача на максимум, или строго убывающая функция в случае задачи минимизации. Если же $\Psi(\xi) \notin \mathrm{Sol}$, то функция приспособленности принимает меньшее значение, чем в любом допустимом решении, что соответствует штрафу за нарушение ограничений задачи. Кроме того, будем предполагать, что в ГА $\Phi(\xi) \geq 0$ для любого $\xi \in \mathcal{B}$.

Построение новой особи (потомка) начинается с выбора пары родительских особей из $\Pi^{(t)}$ при помощи вероятностного оператора селекции $\mathrm{Sel} : \mathcal{B}^N \to \{1, \ldots, N\}$, в котором номера родительских особей выбираются с учетом их приспособленности. К генотипам выбранных особей применяется оператор кроссинговера $\mathrm{Cross} : \mathcal{B} \times \mathcal{B} \to \mathcal{B} \times \mathcal{B}$ (в некоторых генетических алгоритмах $\mathrm{Cross} : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$), заменяющий часть генов од-

ного генотипа генами другого. Полученный генотип подвергается действию оператора мутации $\mathrm{Mut} : \mathcal{B} \to \mathcal{B}$, при этом часть его генов изменяется случайным образом. Здесь и далее под вероятностным оператором понимаем рандомизированную процедуру, такую что при заданном входном аргументе распределение вероятностей на ее выходе не зависит от предшествующей работы алгоритма.

В качестве критерия остановки ГА может использоваться, например, ограничение на число итераций или условие достижения оптимума, когда это возможно установить. Результатом ГА является лучшее построенное решение.

В настоящей работе, наряду с пропорциональной селекцией, исследуется другой широко известный оператор турнирной селекции [199]. При действии данного оператора из популяции равновероятно (с возвращением) выбираются s особей и в качестве родителя используется лучшая по приспособленности среди выбранных. Настраиваемый параметр s называется размером турнира. Параметры N и s, вообще говоря, зависят от индивидуальной задачи и их выбор может существенно влиять на скорость сходимости популяции к решениям приемлемого качества.

Операторы мутации и кроссинговера в ГА с некоторым упрощением моделируют процессы мутации и скрещивания, происходящие в живой природе, где они сводятся к случайным изменениям в последовательности нуклеотидов молекулы ДНК [4]. Степень воздействия кроссинговера и мутации в КГА регулируется вероятностью кроссинговера P_{c} и вероятностью мутации P_{m}. В настоящее время известно множество вариантов генетического алгоритма, имеющих разные операторы селекции, мутации, кроссинговера и несколько различающихся в своих схемах [269].

В КГА на каждой итерации происходит полная замена популяции. Во многих других ГА используется *стационарная стратегия* управления популяцией, где на каждой итерации вычисляются только две или одна

новая особь, заменяющие в популяции особи с наименьшей приспособленностью [269].

Далее приводится описание операторов ЭА и его общая схема. Определения фенотипа, функции представления решений и популяции, введенные выше для ГА, сохраняются без изменений. Функция приспособленности генотипа в случае ЭА может иметь более общий вид, чем в случае ГА. Известны эволюционные алгоритмы как для задач с одним критерием оптимизации, так и для многокритериальных задач, поэтому допускается и многокритериальная вектор-функция приспособленности. В качестве примеров известных ЭА в главе 1 описываются эволюционные стратегии и многокритериальные ЭА.

На практике общие схемы эволюционных алгоритмов зачастую используются в комбинации с некоторыми эвристическими процедурами, учитывающими специфику решаемой задачи. Наиболее часто такими процедурами являются методы локальной оптимизации, жадные алгоритмы корректировки решений и декодирующие эвристики. Специфика задачи в ЭА может быть использована при решении вспомогательных задач математического программирования в операторах кроссинговера и при построении начальной популяции.

В **главе 2** исследуется точность и трудоемкость эволюционных методов и проводится их сопоставление с методами локального поиска.

В разделе 2.1 найдены достаточные условия, при которых генетический алгоритм с полной заменой популяции при турнирной селекции достигает локального оптимума в среднем за полиномиально ограниченное время, и доказано выполнение этих условий на классе задач комбинаторной оптимизации с гарантированными локальными оптимумами (GLO). Предполагается, что двоичное представление решений совпадает с кодировкой решений задачи комбинаторной оптимизации, а при выполнении условия остановки происходит новая инициализация счетчика итераций и

популяции X^0, и вычислительный процесс продолжается. Данный вариант ГА обозначается далее через GA.

Пусть в результате кроссинговера с вероятностью не менее некоторой константы ε, $0 < \varepsilon \leq 1$, образуются особи $(\mathbf{x}', \mathbf{y}') = \mathrm{Cross}(\mathbf{x}, \mathbf{y})$, хотя бы одна из которых не уступает по приспособленности родительским особям $\mathbf{x}, \mathbf{y} \in B$ при любых $\mathbf{x}, \mathbf{y} \in B$, причем константа ε не зависит от I. Для одноточечного кроссинговера последнее неравенство выполняется с $\varepsilon = 1 - P_c$, если $P_c < 1$ – константа, не зависящая от I. Указанное условие также выполняется с $\varepsilon = 1$, если один из двух потомков – решение задачи оптимальной рекомбинации, рассматриваемой далее в главе 3.

Пусть имеется задача $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f_I)$ и задана некоторая система окрестностей $\{\mathcal{N}_I(\mathbf{y}) : \mathbf{y} \in \mathrm{Sol}(I)\}$. Обозначим через $h(I)$ число различных неоптимальных значений целевой функции f_I, т. е. $h(I) = |\{f_I(\mathbf{x}) : \mathbf{x} \in \mathrm{Sol}(I)\}| - 1$. Тогда, начиная с любого решения, локальный поиск достигает локального оптимума не более чем за $h(I)$ итераций, улучшающих значение целевой функции. Пусть $L(I)$ обозначает минимальную вероятность достижения любого заданного решения в пределах окрестности:

$$L(I) = \min_{\mathbf{x} \in \mathrm{Sol}(I),\ \mathbf{x}' \in \mathcal{N}_I(\mathbf{x})} \mathbf{P}\{\mathrm{Mut}(\mathbf{x}) = \mathbf{x}'\}.$$

Чем выше величина $L(I)$, тем больше согласованность оператора мутации Mut с системой окрестностей $\mathcal{N}_I$. Показано, что если X^0 содержит допустимое решение, $s \geq rN$, $r > 0$, $h(I) > 1$, $L(I) > 0$ и

$$N \geq \frac{2(1 + \ln h(I))}{L(I)\varepsilon(1 - 1/e^{2r})},$$

то GA с условием остановки $t = h + 1$ имеет свойства: 1) локальный оптимум посещается к итерации $h(I)$ с вероятностью не менее $1/e$; 2) локальный оптимум достигается не более, чем за $eh(I)$ итераций в среднем.

С использованием свойства 2) получены полиномиальные верхние оценки на среднее время получения оптимума в семействе задач с целевой функцией из класса ONEMAX**, введенного С. Дросте, Т. Йансеном и

И. Вегенером [169], и семействе задач вершинного покрытия графа $G(k)$, предложенного Л.А. Заозерской [91]. Кроме того, из свойства 2) вытекает

Теорема 2.1. *Если задача* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f_I) \in \mathrm{NPO}$ *полиномиально ограничена, функция* $1/L(I)$ *полиномиально ограничена, а популяция* X^0 *содержит допустимое решение, то при соответствующих параметрах GA локальный оптимум впервые достигается в среднем за полиномиально ограниченное время.*

Задача Π из класса NPO принадлежит классу задач с *гарантированными локальными оптимумами* (GLO) [116], если выполняются следующие два условия:

1) по крайней мере одно допустимое решение может быть вычислено за полиномиально ограниченное время для любого входа $I \in \mathrm{Inst}$;

2) существует натуральное число k такое, что все локальные оптимумы задачи Π относительно некоторой k-ограниченной системы окрестностей имеют константную оценку точности.

Примерами задач из класса GLO являются задача максимальной выполнимости логической формулы, задачи о наибольшем независимом множестве, о наименьшем доминирующем множестве вершин и о наименьшем вершинном покрытии в графах со степенью вершин, ограниченной константой, а также задача о максимальном разрезе графа [116].

Теорема 2.1 применима для оценки возможностей ГА отыскивать приближенные решения с априорной оценкой точности на классе GLO.

Следствие 2.1. *Пусть* $\Pi \in \mathrm{GLO}$, $n > k$ *и популяция* X^0 *содержит допустимое решение. Тогда при операторе мутации* Mut^* *и соответствующем выборе параметров GA решение с константной оценкой точности впервые достигается в среднем за полиномиально ограниченное время.*

В разделе 2.2 исследуется динамика численности особей высокой приспособленности в популяции генетического алгоритма с турнирной селек-

цией при отсутствии оператора кроссинговера. Для анализа ГА предлагается математическая модель эволюционного процесса, возникающего при его работе. С использованием данной модели получены оценки числа приближенных решений достаточно высокого качества на заданной итерации ГА и исследована зависимость числа таких решений от размера турнира.

Трудности применения известных моделей ГА (см., например, [251, 276, 289]) к анализу задач комбинаторной оптимизации связаны с большой размерностью матрицы перехода цепи Маркова. Предложенная в данном разделе модель включает в себя меньшее число состояний и позволяет найти нижние и верхние оценки числа особей с приспособленностью не ниже заданного уровня.

Пусть $\Phi_0 = \min_{\xi \in \mathcal{B}} \Phi(\xi)$ и заданы линии уровня функции приспособленности $\Phi_1, \ldots, \Phi_d$, такие что $\Phi_0 < \Phi_1 < \Phi_2 \ldots < \Phi_d$. Будем предполагать, что число линий уровня, а также их значения могут зависеть от индивидуальной задачи I. С выбранными линиями уровня сопоставляются подмножества $H_i = \{\xi : \Phi(\xi) \geq \Phi_i\},\ i = 0, \ldots, d$. Кроме того, полагаем $H_{d+1} = \emptyset$.

Пусть для всех $i = 0, ..., d$ и $j = 1, ..., d$ априори известны нижние α_{ij} и верхние β_{ij} оценки вероятности перехода из $H_i \backslash H_{i+1}$ в H_j при действии оператора мутации, а именно для любого $\xi \in H_i \backslash H_{i+1}$

$$\alpha_{ij} \leq \mathbf{P}\{\mathrm{Mut}(\xi) \in H_j\} \leq \beta_{ij}.$$

Обозначим матрицу с элементами α_{ij}, где $i = 0, ..., d$ и $j = 1, ..., d$ через $\mathbf{A}$; аналогичную матрицу верхних оценок обозначим через $\mathbf{B}$. Представим популяцию на шаге t при помощи *вектора популяции*

$$\mathbf{z}^{(t)} = (z_1^{(t)}, z_2^{(t)}, \ldots, z_d^{(t)}),$$

где $z_i^{(t)} \in \mathbb{R}$ – доля генотипов из H_i в популяции X^t.

Будем называть $(d \times d)$-матрицу с элементами γ_{ij} *монотонной*, если $\gamma_{i-1,j} \leq \gamma_{i,j}$ для всех i, j от 1 до d. Пусть $\mathbf{E}$ – символ математического ожи-

дания; $\mathbf{W}$ – $(d \times d)$-матрица с элементами $w_{ij} = \alpha_{ij} - \alpha_{i-1,j}$; $\mathbf{I}$ – единичная матрица; $\mathbf{a} = (\alpha_{01}, ..., \alpha_{0d})$. Если матрица $\mathbf{A}$ монотонна и $||\mathbf{W}|| < 1$ в некоторой матричной норме, то для любого натурального t имеет место нижняя оценка числа приближенных решений достаточно высокого качества

$$\mathbf{E}[\mathbf{z}^{(t)}] \geq \mathbf{E}[\mathbf{z}^{(0)}]\mathbf{W}^t + \mathbf{a}(\mathbf{I} - \mathbf{W})^{-1}(\mathbf{I} - \mathbf{W}^t).$$

При условии монотонности матрицы $\mathbf{B}$ имеет место оценка сверху

$$\mathbf{E}[z_j^{(t+1)}] \leq \beta_{dj} - \sum_{i=1}^{d}(\beta_{ij} - \beta_{i-1,j})(1 - \mathbf{E}[z_i^{(t)}])^s, \quad j = 1, \ldots, d.$$

Итеративным применением последнего неравенства каждая компонента вектора $\mathbf{E}[\mathbf{z}^{(t)}]$ может быть оценена сверху при любом t.

Если при всех $i = 0, ..., d$, $j = 1, ..., d$, вероятность $\mathbf{P}\{\mathrm{Mut}(\xi) \in H_j\}$ не зависит от выбора $\xi \in H_i \backslash H_{i+1}$, то существуют матрицы оценок $\mathbf{A}$ и $\mathbf{B}$, $\mathbf{A} = \mathbf{B}$. В таком случае оператор мутации будем называть *ступенчатым* относительно набора линий уровня $\Phi_1, ..., \Phi_d$ и обозначать матрицу $\mathbf{A}$ через $\mathbf{M}$.

Ступенчатый оператор мутации будем называть *монотонным*, если матрица $\mathbf{M}$ монотонна. Для таких операторов мутации проведено исследование свойств вектора популяции, связанных с изменением размера турнира при $N \to \infty$. С использованием полученных асимптотических свойств доказана

Теорема 2.2. *Пусть* $\mathbf{z}^{(t)}$ *и* $\hat{\mathbf{z}}^{(t)}$ *– векторы популяции ГА с турнирами величины* s *и* $\hat{s}$ *соответственно, причем* $s \leq \hat{s}$. *Тогда, если ГА имеет монотонный оператор мутации и особи начальных популяций одинаково распределены, то для любых* $t = 0, 1, \ldots$ *и* $i = 1, ..., d$ *при достаточно большом размере популяции имеет место неравенство* $\mathbf{E}[z_i^{(t)}] \leq \mathbf{E}[\hat{z}_i^{(t)}]$.

Выделенный здесь случай монотонного оператора мутации характеризует ситуацию, в которой при $N = 1$ становятся точными нижние оценки,

а при $N \to \infty$ – асимптотически точными верхние оценки на математическое ожидание вектора популяции. Установлено, что для оператора мутации классического ГА условие монотонности во многих случаях эквивалентно условию принадлежности функции Ф к классу строго монотонных псевдобулевых функций [113].

В разделе 2.3 исследуются условия, при которых эволюционная стратегия (1+1)-ES (рандомизированный алгоритм локального поиска) оказывается «наилучшим» методом в классе эволюционных алгоритмов по распределению вероятностей приспособленности лучшего полученного генотипа.

Будем называть *эволюционным алгоритмом с неограниченной памятью* следующий рандомизированный алгоритм: начальная популяция $X^0 \in \mathcal{B}^N$ строится некоторым случайным или детерминированным способом. Последовательность генотипов, построенных в ЭА с неограниченной памятью к началу итерации t, обозначим через σ^{t-1}, а множество всех входящих в σ^{t-1} элементов – через $\mathrm{A}^{(t-1)}$. На каждой итерации t, $t = 1, 2, \ldots$, вычисляется промежуточная популяция X^t из N'' генотипов потомков с помощью оператора воспроизведения $\mathrm{Rep}(\eta^1, \ldots, \eta^{N'})$, где $\eta^1, \ldots, \eta^{N'}$ – некоторые из ранее построенных особей. Родительские особи $\eta^1, \ldots, \eta^{N'}$ на итерации t выбираются с помощью рандомизированного оператора *селекции с неограниченной памятью* $\mathrm{Sel}_\infty : \mathcal{B}^{N+N''(t-1)} \to \mathcal{B}^{N'}$, так что $\mathrm{Sel}_\infty(\sigma^{t-1}) \subseteq \mathrm{A}^{(t-1)}$. Работа ЭА с неограниченной памятью представляет собой итерации следующей композиции случайных отображений:

$$X^t = \mathrm{Rep}(\mathrm{Sel}_\infty(X^0, \ldots, X^{t-1})), \quad t = 1, 2, \ldots.$$

Алгоритм, удовлетворяющий этому определению, обозначим через EA.

Эволюционная стратегия (1+1)-ES является рандомизированным алгоритмом локального поиска и начинает работу с начального генотипа, построенного случайно или детерминированно с помощью некоторой процедуры инициализации. На каждой итерации t, $t = 0, 1, \ldots$, по текущему ге-

нотипу $\zeta^{(t)}$ строится новый генотип ζ' посредством оператора мутации Mut. В случае, если новый генотип по приспособленности не уступает $\zeta^{(t)}$, полагаем $\zeta^{(t+1)} := \zeta'$. В противном случае $\zeta^{(t+1)} := \zeta^{(t)}$.

Обозначим максимум функции приспособленности на последовательности генотипов $\sigma = (\xi_1, \ldots, \xi_k)$ через $\check{\Phi}(\sigma)$, т. е. $\check{\Phi}(\sigma) = \max_{i=1,\ldots,k} \Phi(\xi_i)$. Будем говорить, что оператор воспроизведения Rep *доминируется* оператором мутации Mut, если для любой N'-элементной последовательности генотипов X' и любого $\eta \in \mathcal{B}$, таких что $\Phi(\eta) \geq \check{\Phi}(X')$, при всех $\phi > \Phi(\eta)$:

$$\mathbf{P}\left\{\Phi(\mathrm{Mut}(\eta)) \geq \phi\right\} \geq \mathbf{P}\left\{\check{\Phi}(\mathrm{Rep}(X')) \geq \phi\right\}.$$

Основным результатом данного раздела является следующая теорема о сравнении эволюционных алгоритмов с эволюционной стратегией (1+1)-ES.

Теорема 2.3. *Для любого эволюционного алгоритма EA с неограниченной памятью при* $\Phi(\zeta^{(0)}) \geq \check{\Phi}(X^0)$ *выполняются неравенства*

$$\mathbf{P}\{\Phi(\zeta^{(t)}) \geq \phi\} \geq \mathbf{P}\{\check{\Phi}(\sigma^t) \geq \phi\}, \quad t \in \mathbb{Z}_+, \quad \phi \in \mathbb{R},$$

в том и только в том случае, когда оператор Mut *доминирует оператор* Rep.

Показано, что в условиях применимости данной теоремы (1+1)-ES не уступает никакому другому ЭА в терминах математического ожидания приспособленности лучшего полученного генотипа на любой заданной итерации, а также по среднему времени поиска оптимального генотипа.

В разделе 2.4 предлагаются и обосновываются статистические методы построения доверительных интервалов для числа локальных оптимумов ν на основе результатов многократного выполнения локального поиска. Построены нижние границы для ν, обеспечивающие заданную доверительную вероятность. Кроме того, в предположении равновероятного попадания локального поиска во все локальные оптимумы построены двусторонние доверительные интервалы для ν. Предлагаемые методы предназначены для

исследования индивидуальных задач комбинаторной оптимизации, в которых большая размерность пространства решений не позволяет осуществить полный перебор всех его элементов за приемлемое время.

Оценки числа локальных оптимумов и других параметров множества локальных оптимумов задачи часто оказываются актуальными при исследовании ЭА. Во многих ЭА перед добавлением в популяцию каждый новый генотип проходит процедуру локальной оптимизации. Примеры таких гибридных алгоритмов могут быть найдены в [161, 163] и других работах. При использовании такого подхода возникает вопрос о мощности множества всех генотипов, получаемых в результате локальной оптимизации. Кроме того, существует большое число эвристических методов решения задач комбинаторной оптимизации, в которых локальный поиск используется многократно с различными начальными точками (см., например, [74, 274, 295]). Для успешного применения таких методов важное значение имеет выбор системы окрестностей. Сложность индивидуальной задачи с заданной системой окрестностей характеризуется целым рядом параметров, одним из которых является число локальных оптимумов.

Результаты второй главы опубликованы в [16, 39, 135, 137, 174, 181, 182, 272].

В **главе 3** исследуются сложность и методы решения *задачи оптимальной рекомбинации*, состоящей в отыскании наилучшего возможного результата кроссинговера при заданных двух родительских генотипах, представляющих допустимые решения задачи комбинаторной оптимизации. В настоящей главе ЗОР рассматривается в предположении двоичного представления решений, совпадающего с кодировкой решений задачи комбинаторной оптимизации. В этой главе предполагается, что для рассматриваемых индивидуальных задач множество допустимых решений не пусто.

Определение 3.1. *Задачей оптимальной рекомбинации для задачи комбинаторной оптимизации* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f)$ *является задача комбинаторной оптимизации* $\overline{\Pi} = (\overline{\mathrm{Inst}}, \overline{\mathrm{Sol}}, \overline{f})$, *такая что всякая индивидуальная задача* $\overline{I} \in \overline{\mathrm{Inst}}$ *имеет вид* $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2)$, *где* $I \in \mathrm{Inst}$, $\mathbf{p}^1 = (p^1_1, \ldots, p^1_{n(I)}) \in \mathrm{Sol}(I)$, $\mathbf{p}^2 = (p^2_1, \ldots, p^2_{n(I)}) \in \mathrm{Sol}(I)$, *и*

$$\overline{\mathrm{Sol}}(\overline{I}) = \{\mathbf{x} \in \mathrm{Sol}(I) |\ x_j = p^1_j \text{ или } x_j = p^2_j,\ j = 1, \ldots, n(I)\}. \tag{3.1}$$

Критерий оптимизации в $\overline{I}$ *тот же, что и в* I, *т. е.* $\overline{f}_{\overline{I}} \equiv f_I$.

Допустимые решения $\mathbf{p}^1, \mathbf{p}^2$ задачи I называются *родительскими* решениями для задачи $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2)$. Далее обозначаем $D(\mathbf{p}^1, \mathbf{p}^2) = \{j : p^1_j \neq p^2_j\}$.

Определение 3.1 может интерпретироваться как «фиксация» в ЗОР тех значений генов, в которых оба родительских генотипа совпадают. В литературе, однако, известны и другие постановки задач рекомбинации. Например, в работах [15, 34, 134, 147, 237] хорошие экспериментальные результаты показали ГА, в которых решается задача рекомбинации с «фиксацией» только тех значений генов, в которых оба родительских генотипа имеют значение, равное 0. Такая постановка задачи далее будет называться *ослабленной задачей оптимальной рекомбинации*, и для ее формулировки достаточно заменить условие (3.1) в определении 3.1 на

$$\overline{\mathrm{Sol}}(\overline{I}) = \{\mathbf{x} \in \mathrm{Sol}(I) |\ x_j \leq p^1_j \text{ или } x_j \leq p^2_j,\ j = 1, \ldots, n(I)\}. \tag{3.2}$$

В главе 3 ряд задач комбинаторной оптимизации рассматриваются как задачи булевого линейного программирования:

$$\max \quad f(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j, \tag{3.3}$$

$$\sum_{j=1}^{n} q_{ij} x_j \leq b_i, \quad i = 1, \ldots, m, \tag{3.4}$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n. \tag{3.5}$$

Здесь $\mathbf{x} \in \{0,1\}^n$ – вектор булевых переменных, а исходные данные $c_j, b_i, q_{ij},\ i = 1,\ldots,m,\ j = 1,\ldots,n$ – рациональные числа.

В разделе 3.2 с использованием полиномиальных сводимостей между задачами оптимальной рекомбинации доказывается, что ЗОР полиномиально разрешима для задач упаковки множества наибольшего веса, разбиения множества минимального веса и одного варианта простейшей задачи размещения предприятий в постановке булевого линейного программирования. Для задачи булевого линейного программирования (3.3)-(3.5) введем обозначение $U_i = \{j \mid q_{ij} \neq 0,\ j = 1,\ldots,n\},\ i = 1,\ldots,m.$

Теорема 3.2. *ЗОР для задачи булевого линейного программирования (3.3)-(3.5) сводится к задаче о независимом множестве в 2-раскрашиваемом гиперграфе, где 2-раскраска является частью входных данных. Каждое ребро в получаемом гиперграфе содержит не более чем $U_{\max}$ вершин, где $U_{\max} = \max_{i=1,\ldots,m} |U_i|$, и трудоемкость данной сводимости есть $O(m(2^{U_{\max}} + n))$.*

Из теоремы 3.2 следует существование полиномиально вычислимого оператора оптимальной рекомбинации для задач булевого линейного программирования, когда в каждое ограничение входят не более двух переменных. Таким образом, получено обобщение результата [122] об эффективной разрешимости ЗОР для задачи о клике наибольшего веса.

В разделе 3.3 показана сложность ряда задач оптимальной рекомбинации. Доказана NP-трудность оптимальной рекомбинации для задач булевого линейного программирования с тремя переменными в каждом ограничении. Далее рассмотрена ЗОР для задач об одномерном булевом рюкзаке и одномерной упаковке в контейнеры в постановке булевого линейного программирования. Задача об одномерном булевом рюкзаке имеет формулировку

$$\max\left\{\sum_{j=1}^{n} c_j x_j \mid \sum_{j=1}^{n} a_j x_j \leq A,\ x_j \in \{0,1\},\ j = 1,\ldots,n\right\},$$

где $a_j \geq 0, c_j \geq 0, j = 1, \ldots, n$, и $A \geq 0$ целочислены.

В задаче об одномерной упаковке в контейнеры указан размер контейнера A и целые числа $a_1, \ldots, a_k$, представляющие собой размеры предметов. Требуется разместить предметы в минимальное число контейнеров так, чтобы сумма размеров предметов в каждом контейнере не превышала A.

Пусть булева переменная y_j является индикатором использования контейнера с номером j, $j = 1, \ldots, k$, а булева переменная x_{ij} – индикатором упаковки предмета i в контейнер j при $i, j = 1, \ldots, k$. Требуется найти

$$\min \sum_{j=1}^{k} y_j,$$

$$\sum_{j=1}^{k} x_{ij} = 1, \quad i = 1, \ldots, k,$$

$$\sum_{i=1}^{k} a_i x_{ij} \leq A, \quad j = 1, 2, \ldots, k,$$

$$y_j \geq x_{ij}, \quad i = 1, \ldots, k, \ j = 1, \ldots, k,$$

$$x_{ij}, \ y_j \in \{0, 1\}, \quad i = 1, \ldots, k, \ j = 1, 2, \ldots, k.$$

Пусть решения задачи об одномерной упаковке в контейнеры кодируются посредством $(k \times k)$-матрицы значений x_{ij} (соответствующий вектор $(y_1, \ldots, y_k)$ вычисляется однозначно). Тогда справедлива

Теорема 3.3. *ЗОР и ослабленная ЗОР для одномерных задач о булевом рюкзаке и об упаковке в контейнеры являются NP-трудными.*

Рассмотрим задачу коммивояжера (ЗК): дан ориентированный граф G без петель и кратных дуг с множеством вершин V и множеством

дуг $\mathcal{A}$, где длина дуги $(i,j) \in \mathcal{A}$ равна $c_{ij} \geq 0$. Найти гамильтонов контур минимальной длины.

Если для любой дуги $(i,j) \in \mathcal{A}$ существует обратная, и $c_{ij} = c_{ji}$, то ЗК называется *симметрической* и граф G можно считать неориентированным. Если же такое свойство не предполагается, то имеет место *общий случай* ЗК.

Допустимое решение задачи коммивояжера в ЭА может кодироваться как строка, в которой последовательно записаны все компоненты матрицы перестановки, отвечающей маршруту коммивояжера. В таком случае ЗОР состоит в поиске кратчайшего маршрута коммивояжера, совпадающего с двумя родительскими допустимыми решениями по тем дугам (или ребрам), по которым проходят оба родительских решения, и не проходящего по дугам (или ребрам), отсутствующим в обоих из них. NP-трудность ЗОР в симметрическом случае ЗК доказана с использованием результатов А. Итаи, Х. Пападимитриу и Дж. Шварцфитера [217] об NP-полноте задачи проверки свойства гамильтоновости решеточных графов.

Теорема 3.4. *ЗОР для задачи коммивояжера в симметрическом случае NP-трудна в сильном смысле.*

Аналогично доказывается теорема 3.5 об NP-трудности ЗОР для задачи отыскания кратчайшей гамильтоновой цепи в графе.

В общем случае ЗК задача оптимальной рекомбинации не является обобщением рассмотренной в теореме 3.4. Даже при симметрической матрице расстояний (c_{ij}) пара родительских «маршрутов», понимаемых как контуры, обусловливает иное множество допустимых решений ЗОР, чем та же пара «маршрутов», понимаемых как циклы. Таким образом, общий случай требует специального анализа сложности. С использованием известной сводимости задачи вершинного покрытия к ЗК доказана

Теорема 3.6. *ЗОР для задачи коммивояжера в общем случае NP-трудна в сильном смысле.*

Рассмотренные выше задачи оптимальной рекомбинации сводятся к ЗК с предписанными ребрами на графах, где степень вершин равна не более 4 в симметрическом и не более 3 – в общем случае. Решение таких задач может быть получено алгоритмами, предложенными Д. Эппштейном [171]. В частности, трудоемкость решения ЗОР для общего случая ЗК составляет $O(|V| \cdot 1.42^{|V|})$.

В работе [42] рассмотрен другой подход к кодировке решений для задач на перестановках и соответствующая ему ЗОР. Как показано в [43], в случае задачи о кратчайшем гамильтоновом пути этот способ представления решений также приводит к NP-трудной ЗОР, однако для «почти всех» пар родительских решений ЗОР полиномиально разрешима.

Результаты третьей главы опубликованы в [29, 38, 175, 176].

В **главе 4** предложены эволюционные алгоритмы с возможностями методов динамического программирования (ДП).

В разделе 4.1 формулируется достаточно общая схема динамического программирования. С целью единообразного описания алгоритмов ДП здесь вводится дополнительное предположение о том, что решаемая задача комбинаторной оптимизации Π может быть преобразована во вспомогательную задачу многокритериальной оптимизации специального вида Π', согласованную с общей схемой ДП.

Пусть имеется d' критериев оптимизации. Индивидуальная задача I' из многокритериальной задачи Π' определяется четверкой $(d', g, \mathcal{S}, \mathcal{D})$. Здесь $\mathcal{S}$ – *пространство поиска*, $g: \mathcal{S} \to (\mathbb{R} \cup \{\infty\})^{d'}$ – векторно-значная целевая функция, и $\mathcal{D} \subseteq \mathcal{S}$ – множество допустимых решений. Введем частичный порядок $\succeq$ для обозначения доминирования по Парето: $(y'_1, \ldots, y'_{d'}) \succeq (y_1, \ldots, y_{d'})$ тогда и только тогда, когда $y'_j \leq y_j$ для всех j, таких что g_j – критерий минимизации, и $y'_j \geq y_j$ для всех j, таких что g_j – критерий максимизации ($\mathbf{y}' \succ \mathbf{y}$, если $\mathbf{y}' \succeq \mathbf{y}$ и $\mathbf{y} \not\succeq \mathbf{y}'$). Решение задачи I' состоит в отыскании полного множества альтернатив (ПМА).

Далее предполагается, что ПМА для задачи I' с помощью некоторой эффективной процедуры (например, метода обратного хода) преобразуется в оптимальное решение $\mathbf{x}^* \in \mathrm{Sol}(I)$ индивидуальной задачи I из П.

Алгоритм ДП для задачи Π' выполняется за $\bar{n}$ итераций, называемых *стадиями*. На каждой стадии i вычисляется и записывается в память некоторое множество *состояний* $\mathcal{S}_i \subseteq \mathcal{S}$. Будем говорить, что алгоритм решает индивидуальную задачу I', если множество состояний, вычисленных на последней стадии ДП, представляет собой ПМА для I'.

Пусть имеются $\bar{n}$ конечных множеств отображений $\mathcal{F}_i,\ i = 1, \ldots, \bar{n}$, состоящих из переходных функций вида $F\colon \mathcal{S} \to \mathcal{S}'$. Здесь $\mathcal{S}'$ является расширением пространства $\mathcal{S}$ или совпадает с ним. С целью исключения элементов, принадлежащих $\mathcal{S}'\backslash\mathcal{S}$, на каждой стадии i используется семейство функционалов H_F, $H_F\colon \mathcal{S}' \to \mathbb{R}$, таких что при любом $F \in \mathcal{F}_i$ выполнено $F(S) \in \mathcal{S}$ тогда и только тогда, когда $H_F(S) \leq 0$.

Сначала рассматривается *упрощенный* алгоритм ДП, в котором на этапе инициализации формируется множество $\mathcal{S}_0$, являющееся конечным подмножеством $\mathcal{S}$. На i-той стадии вычисляется множество состояний $\mathcal{S}_i$:

$$\mathcal{S}_i = \{F(S) \mid S \in \mathcal{S}_{i-1},\ F \in \mathcal{F}_i,\ H_F(S) \leq 0\}.$$

Результатом работы упрощенного алгоритма ДП является минимальное по включению подмножество $\mathcal{S}'_{\bar{n}} \subseteq \mathcal{S}_{\bar{n}}$, такое что $g(\mathcal{S}'_{\bar{n}}) = g(\mathcal{S}_{\bar{n}})$.

Для снижения трудоемкости в приложениях ДП, как правило, используется принцип оптимальности Р. Беллмана [130] или его обобщения, что позволяет удалить из рассмотрения неперспективные состояния, не нарушая оптимальности результата. Во многокритериальном случае классический принцип Р. Беллмана, как правило, неприменим, однако неперспективные состояния могут быть удалены с помощью соответствующего отношения доминирования на множестве состояний ДП [79, 228, 293].

Рассмотрим предпорядок $\preceq_{\mathrm{dom}}$ на $\mathcal{S}'$, такой что $S \preceq_{\mathrm{dom}} S'$ тогда и только тогда, когда $g(S) \preceq g(S')$ или $S \notin \mathcal{S}$, $S' \in \mathcal{S}$. Установлено, что два

следующих условия позволяют корректно использовать отношение $\preceq_{\text{dom}}$ для исключения неперспективных состояний в ДП.

С.1. При любых $S, S' \in \mathcal{S}$, $i = 1, \ldots, \bar{n}$, если $S \preceq_{\text{dom}} S'$, то $F(S) \preceq_{\text{dom}} F(S')$ для всех $F \in \mathcal{F}_i$ таких, что $F(S') \in \mathcal{S}$.

С.2. Для любых $S, S' \in \mathcal{S}$, $i = 1, \ldots, \bar{n}$ и $F \in \mathcal{F}_i$, если $S \preceq_{\text{dom}} S'$, то $H_F(S') \leq H_F(S)$.

При выполненных условиях С.1 и С.2 в ДП достаточно хранить не все множество $\mathcal{S}_i$, а лишь минимальное по включению его подмножество, доминирующее $\mathcal{S}_i$, и общая схема ДП представима в следующем виде.

Алгоритм 4.1. Динамическое программирование для задачи Π'

1. $\mathcal{T}_0 := \mathcal{S}_0$.
2. Для всех $i = 1 \ldots \bar{n}$:
3. $\quad \mathcal{T}_i := \emptyset$.
4. $\quad$ Для всех $S \in \mathcal{T}_{i-1}$ & $F \in \mathcal{F}_i$:
5. $\quad\quad$ Если $H_F(S) \leq 0$ & $\nexists S' \in \mathcal{T}_i$: $F(S) \preceq_{\text{dom}} S'$, то
6. $\quad\quad\quad \mathcal{T}_i := (\mathcal{T}_i \setminus \{S' \in \mathcal{T}_i \mid S' \prec_{\text{dom}} F(S)\}) \cup \{F(S)\}$.
7. $\quad$ Конец цикла.
8. Конец цикла.
9. Результат: $\mathcal{T}_{\bar{n}}$.

Если упрощенный алгоритм ДП вычисляет множество состояний $\mathcal{S}'_{\bar{n}}$, представляющее собой ПМА для I', и условия С.1 и С.2 выполнены, то алгоритм 4.1 также находит ПМА для I'.

В разделе 4.2 при условии применимости к задаче Π' алгоритма ДП предлагается многокритериальный ЭА с аналогичными свойствами. Для выявления эффектов, связанных со спецификой ЭА, необходимо ввести дополнительные ограничения на трудоемкость каждого из операторов многокритериального ЭА. Такой подход к сравнению ЭА и ДП реализован в следующей теореме 4.1. Обозначим через T_{DP} трудоемкость алгоритма ДП, а через $W_{\max}$ – наибольшую из мощностей множеств состояний, вы-

численных за все время работы ДП. Под *обработкой состояния* в ДП будем понимать итерацию внутреннего цикла, т. е. строки 5 и 6 алгоритма 4.1.

Теорема 4.1. *Пусть задача* Π' *разрешима алгоритмом ДП, условия С.1, С.2 выполнены, а обработка состояния в ДП имеет трудоемкость* $\Theta(1)$. *Тогда существует многокритериальный ЭА, где трудоемкость каждого оператора есть* $O(1)$, *и ПМА задачи* Π' *вычисляется в среднем за время* $O(T_{\mathrm{DP}}\bar{n}\log W_{\max})$.

Подобный результат имеет место и в более общем случае, когда время обработки состояния ДП и трудоемкости операторов ЭА полиномиально ограничены. В этом случае верхняя оценка средней трудоемкости многокритериального ЭА отличается от трудоемкости алгоритма динамического программирования на множитель, ограниченный полиномом от длины входных данных и числа состояний ДП.

В разделе 4.3 установлено (теорема 4.2), что для класса задач, удовлетворяющих условиям существования вполне полиномиальной аппроксимационной схемы (FPTAS) Г. Воегингера, на основе многокритериального ЭА может быть построено семейство алгоритмов, являющееся вполне полиномиальной рандомизированной аппроксимационной схемой (FPRAS).

Результаты четвертой главы опубликованы в [35–37, 155, 156].

Глава 5 посвящена применению оптимальной рекомбинации в ГА.

В разделе 5.1 разработаны и исследованы генетический алгоритм и точный гибридный алгоритм для решения задачи наименьшего покрытия множества (ЗНП). Эта задача имеет следующую постановку. Пусть даны множество $M = \{1, ..., m\}$ и набор его подмножеств $M_j \subseteq M$, где $j \in U = \{1, ..., n\}$. Подмножество $J \subseteq U$ называется *покрытием* M, если $\bigcup_{j\in J} M_j = M$. Каждому M_j приписан вес $c_j > 0$. Требуется найти покрытие минимального суммарного веса. ЗНП NP-трудна в сильном смысле и для нее не существует полиномиального алгоритма, вычисляющего решение, не более чем в константу раз превышающее оптимум, если P≠NP.

Для ЗНП предлагается новый вариант ГА (GANP) с недвоичным представлением решений и оператором ЛП-кроссинговера, при действии которого ослабленная ЗОР решается с использованием методов линейного программирования. Кроме того, в данном разделе описывается гибридный алгоритм для решения ЗНП, который представляет собой комбинацию метода перебора L-классов с GANP и эвристикой лагранжевой релаксации. ГА используется для нахождения начального решения, а с помощью метода лагранжевой релаксации строятся нижние и верхние оценки целевой функции на начальном этапе и на подзадачах, возникающих в процессе перебора L-классов.

При решении задач библиотеки OR-Library предложенный ГА показал результаты, не уступающие результатам ГА с двоичным представлением Дж. Бисли и П. Чу [127]. В частности, для всех 50 тестовых задач со случайными данными при размерности до 5000 переменных данным ГА найдены оптимальные решения. Кроме того, для двух индивидуальных задач, связанных с вопросом П. Эрдеша о раскраске гиперграфа, получены решения, улучшающие известные из литературы оценки оптимума.

Эксперимент показал, что применение ЛП-кроссинговера в GANP целесообразно при решении задач, где разрыв двойственности не велик. В противном случае предпочтительнее равномерный кроссинговер. Гибридный алгоритм тестировался на задачах со случайными данными и показал существенное ускорение по сравнению с перебором L-классов [51].

В разделе 5.2 рассматривается *задача управления поставками продукции*, состоящая в минимизации стоимости доставки продукции от множества поставщиков до множества потребителей. Допустимый размер каждой открытой поставки ограничен снизу и сверху, размер потребления для каждого потребителя ограничен снизу, а функции стоимости поставки ли-

нейны при ненулевых объемах поставки. Задача имеет постановку:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij}a_{ij} + c_{ij}x_{ij},$$

$$\sum_{i=1}^{m} x_{ij} \geq A_j,\ j = 1, ..., n,$$

$$\sum_{j=1}^{n} x_{ij} \leq M_i,\ i = 1, ..., m,$$

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

$$m_{ij}z_{ij} \leq x_{ij} \leq M_i z_{ij},\ i = 1, ..., m,\ j = 1, ..., n,$$

где m – число поставщиков; n – число потребителей; переменные z_{ij} являются индикаторами наличия поставки от поставщика i потребителю j, а переменные x_{ij} обозначают размер соответствующей поставки. A_j – минимальное количество продукта, требуемое потребителю j; m_{ij} – минимальное количество продукта, которое поставщик i готов доставить потребителю j; M_i – максимальное общее количество продукта, которое поставщик i может доставить. В целевую функцию входят фиксированные доплаты a_{ij}, связанные с открытием поставки (i, j) и стоимости транспортировки единицы продукции c_{ij}. Параметры $a_{ij}, c_{ij}, m_{ij}, M_i, A_j$ целочисленны и неотрицательны при всех i, j.

Теорема 5.1. *Нахождение $p(m, n)$-приближенного решения задачи управления поставками продукции при любом полиноме $p(m, n)$ представляет собой NP-трудную в сильном смысле задачу.*

Для случая $n = 1$ в данном разделе предложен жадный алгоритм трудоемкости $O(m^2)$ и доказано, что он является 2-приближенным алгоритмом, если стоимость поставки от каждого поставщика i является вогнутой неубывающей функцией на отрезке $[0, M_i]$ (теорема 5.2). В [41] нами предложена FPTAS для случая $n = 1$ в более общей постановке, где предполагается, что объем поставки от каждого поставщика принадлежит объединению конечного числа отрезков, функции стоимости поставок являются

вогнутыми и неубывающими в пределах каждого интервала. В [144, 250] построена FPTAS для более широкого класса целевых функций.

Для рассматриваемой задачи предлагаются два гибридных генетических алгоритма и проводится их исследование и сравнение. Первый из предложенных ГА, обозначаемый через GAgrd, основывается на декодировании генотипов с использованием жадного 2-приближенного алгоритма.

Во втором предложенном ГА, обозначаемом через GAmipr, генотип представляет собой $(m \times n)$-матрицу $\mathbf{Z} = (z_{ij})$. Если при заданной $\mathbf{Z}$ допустимое решение существует, то объемы поставок x_{ij} вычисляются посредством решения транспортной задачи с нижними и верхними ограничениями на объем каждой перевозки. Если же такого решения не существует, то для оценки приспособленности используется штрафная функция.

В GAmipr разработан новый оператор ЦЛП-кроссинговера, в котором перед решением ЗОР в родительские генотипы вносятся случайные изменения. После этого решается (ослабленная) ЗОР для задачи управления поставками продукции в приведенной выше постановке частично целочисленного линейного программирования (ЧЦЛП). Из теоремы 3.3 вытекает, что ЗОР и ослабленная ЗОР для задачи управления поставками продукции являются NP-трудными даже в случае одного потребителя. Оператор ЦЛП-кроссинговера реализован с использованием пакета CPLEX.

Вычислительный эксперимент показал, что GAmipr имеет преимущество по качеству получаемых решений в сравнении с GAgrd и CPLEX.

В разделе 5.3 рассматривается NP-трудная задача балансировки автоматизированной производственной линии и предлагается ГА с оператором ЦЛП-кроссинговера для ее решения. Данная задача имеет ряд отличий от задачи балансировки автоматической сборочной линии [124,129,192], таких как параметризованное время выполнения операции, нестрогие отношения предшествования и параллельное выполнение операций.

В предлагаемом ГА применяется стационарная стратегия управле-

ния популяцией и турнирная селекция. В качестве генотипа используется совокупность булевых переменных в терминах модели [165], кодирующая назначение операций на блоки и станции. Генотипы начальной популяции формируются случайным образом посредством N-кратного применения эвристики GRASP, также реализованной с использованием модели ЧЦЛП.

Действие ЦЛП-кроссинговера состоит в решении ЗОР, получаемой фиксацией значений части булевых переменных в исходной задаче. Как следует из теоремы 3.3, данная ЗОР является NP-трудной задачей.

В вычислительном эксперименте с предложенным ГА сравнивались эвристические и точные алгоритмы из [167, 207, 207], а также эвристика GRASP и метод ветвей и границ с отсечениями из пакета CPLEX.

Эксперименты [160, 209] показали перспективность использования ЗОР в ГА для рассматриваемой задачи и позволили выявить классы задач, на которых ГА с ЦЛП-кроссинговером имеет преимущество перед другими известными алгоритмами. Важной отличительной особенностью данного подхода является возможность его применения к широкому классу задач за счет использования гибких средств построения моделей ЧЦЛП и универсальных пакетов программ для решения такого рода задач.

Результаты пятой главы дают экспериментальное подтверждение целесообразности решения (точного или приближенного) ЗОР в операторах кроссинговера генетических алгоритмов даже в тех случаях, когда ЗОР является NP-трудной задачей. Результаты пятой главы опубликованы в работах [34, 40, 44, 133, 134, 160, 173].

В **заключении** приведены основные результаты диссертации.

Автор благодарит д.ф.-м.н., профессора Александра Александровича Колоколова за полезные советы при выполнении данной работы.

1. Постановки задач и схемы эволюционных алгоритмов

Настоящая глава содержит постановки задач комбинаторной оптимизации и общие схемы эволюционных алгоритмов. Приведенные здесь определения и схемы алгоритмов многократно используются далее в различных разделах диссертации.

1.1. Задачи комбинаторной оптимизации

1.1.1. Базовые определения

Пусть $\{0,1\}^*$ обозначает множество всевозможных строк из нулей и единиц произвольной длины, а $\mathbb{R}$, $\mathbb{Q}$ и $\mathbb{N}$ – множества вещественных, рациональных и натуральных чисел. Для $S \in \{0,1\}^*$ символом $|S|$ обозначается длина строки S.

Определение 1.1. *Задача комбинаторной оптимизации – это тройка* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f_I)$, *где*

1) $\mathrm{Inst} \subseteq \{0,1\}^*$ – *множество индивидуальных задач из* Π;

2) $\mathrm{Sol}(I) \subseteq \{0,1\}^{n(I)}$ – *множество допустимых решений индивидуальной задачи* $I \in \mathrm{Inst}$, $n(I)$ – *размерность пространства решений;*

3) *для каждой* $I \in \mathrm{Inst}$ *определена функция* $f_I : \mathrm{Sol}(I) \to \mathbb{R}$, *которую требуется максимизировать (если* Π – *задача максимизации) или минимизировать (если* Π – *задача минимизации).*

В литературе задачи данного класса именуются также комбинаторными задачами оптимизации [86].

Далее через f_I^* обозначается оптимальное решение индивидуальной задачи I, т. е. $f_I^* = \max\{f_I(\mathbf{x}) : \mathbf{x} \in \mathrm{Sol}(I)\}$, если Π – задача максимизации, либо $f_I^* = \min\{f_I(\mathbf{x}) : \mathbf{x} \in \mathrm{Sol}(I)\}$, если Π – задача минимизации.

Величина $a > 0$ называется полиномиально ограниченной относительно величины $b > 0$, если существует полином с положительными коэффициентами относительно b, ограничивающий сверху значения a. В тех случаях, когда ясно, о какой индивидуальной задаче I идет речь, по умолчанию под *полиномиально ограниченной* величиной будем подразумевать величину, полиномиально ограниченную относительно $|I|$. Функции и алгоритмы будем называть эффективно вычислимыми, если время их вычисления полиномиально ограничено.

Наибольший теоретический интерес представляют задачи комбинаторной оптимизации из класса NPO [140], также известные как *задачи* NP *оптимизации*, в которых вводятся следующие «технические» предположения.

Определение 1.2. *Задача комбинаторной оптимизации принадлежит классу* NPO, *если отношения* $I \in \mathrm{Inst}$ *и* $\mathbf{x} \in \mathrm{Sol}(I)$ *могут быть проверены за полиномиально ограниченное время, размерность* $n(I)$ *полиномиально ограничена, а функция* $f_I : \mathrm{Sol}(I) \to \mathbb{N}$ *полиномиально вычислима для любой* $I \in \mathrm{Inst}$.

Класс NPO является аналогом класса NP для задач комбинаторной оптимизации (подробнее см., например, [140]).

Определение 1.3. *Задача комбинаторной оптимизации называется полиномиально ограниченной, если существует полином от* $|I|$, *ограничивающий значения* $f_I(\mathbf{x})$, $\mathbf{x} \in \mathrm{Sol}(I)$.

Из контекста, как правило, ясно, о какой индивидуальной задаче I идет речь, поэтому для краткости символ I в таких случаях опускается.

В оценках трудоемкости алгоритмов подразумевается вычислительная сложность машины RAM с произвольным доступом к памяти [6], где стандартные арифметические операции имеют константную длительность.

Символы $O(\cdot), \Omega(\cdot)$ и $\Theta(\cdot)$ далее используются в общепринятом смысле (см., например, [70]). $\mathbf{E}$ – символ математического ожидания.

Граф (ориентированный граф) будем обозначать как пару $(\cdot, \cdot)$, где на первом месте множество вершин, а на втором – множество ребер (дуг).

1.1.2. Некоторые классические задачи комбинаторной оптимизации

Рассмотрим несколько классических задач комбинаторной оптимизации в качестве примеров. Способы записи исходных данных при этом не будут уточняться, однако всюду будет предполагаться использование «разумных схем кодирования» (см., например, [27]).

Задача об одномерном булевом рюкзаке. Вход задачи составляют параметры n предметов: для кажого предмета $i,\ i = 1, \ldots, n$, указан его целочисленный вес $a_i \geq 0$ и полезность $c_i \geq 0$, $i = 1, \ldots, n$. Кроме того, задан максимальный предельно допустимый целочисленный вес $A \geq 0$. Задача состоит в отыскании набора предметов $K \subseteq \{1, \ldots, n\}$, максимизирующего суммарную полезность $\sum_{i \in K} c_i$, при условии $\sum_{i \in K} a_i \leq A$.

Для кодировки решений данной задачи будет использоваться наиболее естественный способ, при котором размерность пространства решений совпадает с числом предметов и каждому предмету $i,\ i = 1, \ldots, n$, сопоставлена компонента x_i булева вектора $\mathbf{x}$. Предполагается, что $x_i = 1$, если предмет i выбран, иначе $x_i = 0$.

Задача коммивояжера (ЗК). Дан ориентированный граф без петель и кратных дуг с множеством вершин $V = \{v_1, \ldots, v_{|V|}\}$ и множеством дуг $\mathcal{A}$.

Указаны неотрицательные длины (веса) всех дуг $(v_i, v_j) \in \mathcal{A}$, обозначаемые через $c_{ij} \geq 0$. Требуется найти гамильтонов контур минимальной длины. В случае, когда матрица (c_{ij}) симметрическая, задача коммивояжера называется *симметрической*. Если же такое свойство не предполагается, будем говорить, что имеет место *общий случай* задачи коммивояжера.

В общем случае маршрутом коммивояжера будем называть гамильтонов контур. В симметрическом же случае направление обхода не имеет значения, поэтому маршрутом коммивояжера будем называть гамильтонов цикл в полном неориентированном графе с тем же множеством вершин V и множеством ребер E, где длина ребра (i, j) есть $c_{ij} = c_{ji}$.

Элементы множества $\mathrm{Sol}(I)$ в общем случае ЗК могут состоять, например, из последовательно записанных в двоичной системе счисления номеров вершин маршрута коммивояжера или представлять собой последовательно записанные компоненты матрицы перестановки, отвечающей такому маршруту. Второй из вариантов предпочтительнее тем, что матрицей перестановки любой гамильтонов контур в полном графе представляется единственным образом. По умолчанию далее будет предполагаться использование второго способа кодировки решений.

В симметрическом случае маршрут коммивояжера единственным образом определяется верхнетреугольной матрицей перестановки. Такой способ кодировки решений будет предполагаться далее по умолчанию.

Задача о наименьшем покрытии (ЗНП). Пусть даны базовое множество $M = \{1, ..., m\}$ и набор его подмножеств $M_j \subseteq M,\ j \in U$, где множество индексов $U = \{1, ..., n\}$. Подмножество $J \subseteq U$ называется покрытием M, если $\bigcup\limits_{j \in J} M_j = M$. Каждому $M_j,\ j \in U$ приписана стоимость $c_j > 0$. Требуется найти покрытие минимальной суммарной стоимости. ЗНП называется *невзвешенной*, если предполагается, что все стоимости c_j равны единице.

ЗНП относится к числу NP-трудных задач. Один из первых результатов относительно сложности ЗНП был получен Ю.И. Журавлевым [47], который установил, что в классе локальных алгоритмов конечного индекса задача о вхождении подмножества M_j в какое-либо оптимальное покрытие не разрешима. К ЗНП сводятся многие другие известные задачи дискретной оптимизации: задачи стандартизации, упаковки и разбиения множества [232], задача о наибольшей клике [27] и др. Известна также и обратная сводимость ЗНП к этим задачам [12, 27, 232]. Из сложности ЗНП следует труднорешаемость целого ряда задач (см., например, [12, 27, 86, 226]).

На практике ЗНП возникает при размещении пунктов обслуживания [82, 85], в системах информационного поиска и распознавания образов [30], при назначении экипажей на транспорте [141], в проектировании интегральных схем [50] и т.д.

Задача о вершинном покрытии. Задача о вершинном покрытии (ЗВП) представляет собой NP-трудный частный случай ЗНП (см., например, [27]). Данная задача имеет следующую постановку.

Пусть дан граф $G = (V, E)$ с множеством вершин V и множеством ребер E и указаны положительные веса вершин $c_v, \quad v \in V$. Подмножество $C \subseteq V$ называется *вершинным покрытием* в G, если каждое ребро инцидентно хотя бы одной вершине из C. Требуется найти вершинное покрытие C^* минимального суммарного веса.

Простейшая задача размещения. Пусть имеется возможность открыть n предприятий, каждое из которых может обслуживать любого из m клиентов. Открытие i-го предприятия, $i = 1, ..., n$, обходится в $C_i \geq 0$ единиц стоимости, а производственно-транспортные расходы, связанные с обслуживанием j-го клиента, $j = 1, ..., m$, предприятием i, $i = 1, ..., n$, со-

ставляют $c_{ij} \geq 0$ единиц стоимости. Требуется минимизировать

$$f_{\text{splp}}(\mathbf{x}) = \sum_{i=1}^{n} C_i x_i + \sum_{j=1}^{m} \min_{i:x_i=1} c_{ij}, \tag{1.1}$$

при условии, что

$$\sum_{i=1}^{n} x_i \geq 1, \tag{1.2}$$

где $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ – вектор переменных. Оптимальное решение $\mathbf{x}^*$ представляет собой вектор – индикатор подмножества открытых предприятий. По вектору $\mathbf{x}^*$ легко определяются прикрепления клиентов к открытым предприятиям, обеспечивающие минимальные транспортные расходы.

Данная задача является NP-трудной в сильном смысле, даже если матрица (c_{ij}) удовлетворяет неравенству треугольника [230]. Взаимосвязь простейшей задачи размещения производства с другими задачами, а также некоторые ее приложения описаны в [11, 12, 232].

1.1.3. Поиск приближенных и локально оптимальных решений

Существование эффективных алгоритмов для NP-трудных задач комбинаторной оптимизации не представляется возможным, если справедлива гипотеза о неравенстве классов P и NP (см., например, [27]). В связи с этим с целью эффективного отыскания некоторого «приемлемого» решения от исходной постановки задачи комбинаторной оптимизации зачастую переходят к связанным с ней задачам поиска приближенного решения или локально оптимального решения.

Задача поиска приближенного решения. Введем стандартные определения, формализующие понятие приближенного решения задачи.

Определение 1.4. *Пусть* Π – *задача комбинаторной оптимизации,* $I \in \text{Inst}$ – *ее индивидуальная задача. Оценкой точности решения*

$\mathbf{x} \in \mathrm{Sol}(I)$, $f_I(\mathbf{x}) > 0$, *назовем величину* $R(I, \mathbf{x}) = f_I^*/f_I(\mathbf{x})$, *если* Π – *задача максимизации, или величину* $R(I, \mathbf{x}) = f_I(\mathbf{x})/f_I^*$, *если* Π – *задача минимизации.*

Определение 1.5. *Пусть для* Π *задана функция* $\rho : \mathrm{Inst} \to [1, \infty)$. *Тогда алгоритм, который за полиномиальное время для любой индивидуальной задачи* $I \in \mathrm{Inst}$ *при* $\mathrm{Sol}(I) \neq \emptyset$ *находит допустимое решение* $\mathbf{y}(I)$, $R(I, \mathbf{y}(I)) \leq \rho(I)$, *называется* ρ*-приближенным алгоритмом. Решение* $\mathbf{y}(I)$ *в таком случае называется* ρ*-приближенным решением.*

Получение ρ-приближенного решения для любой $I \in \mathrm{Inst}$ (при условии, что $\mathrm{Sol}(I) \neq \emptyset$) называется *аппроксимацией* Π *с оценкой* ρ.

Определение 1.6. *Семейство* $(1+\varepsilon)$*-приближенных алгоритмов с временной сложностью, полиномиально зависящей от длины входа задачи и от* $1/\varepsilon$ *при любом* $0 < \varepsilon < 1$, *называется* вполне полиномиальной аппроксимационной схемой *(FPTAS).*

Задача поиска локального оптимума. Пусть для всякого элемента $\mathbf{y} \in \mathrm{Sol}(I)$ определена некоторая его окрестность $\mathcal{N}_I(\mathbf{y}) \subseteq \mathrm{Sol}(I)$. Совокупность $\{\mathcal{N}_I(\mathbf{y}) : \mathbf{y} \in \mathrm{Sol}(I)\}$ называется *системой окрестностей.*

Определение 1.7. *Если для* $\mathbf{x} \in \mathrm{Sol}(I)$ *при всяком* $\mathbf{y} \in \mathcal{N}_I(\mathbf{x})$ *выполняется неравенство* $f_I(\mathbf{y}) \leq f_I(\mathbf{x})$ *в случае задачи максимизации или* $f_I(\mathbf{y}) \geq f_I(\mathbf{x})$ *в случае задачи минимизации, то решение* $\mathbf{x}$ *называется локальным оптимумом в системе окрестностей* $\mathcal{N}_I$.

Если $\mathcal{D}(\cdot, \cdot)$ – метрика, заданная для всех элементов $\mathbf{x}, \mathbf{y} \in \mathrm{Sol}(I)$, то $\mathcal{N}_I(\mathbf{x}) = \{\mathbf{y} : \mathcal{D}(\mathbf{x}, \mathbf{y}) \leq k\}$, $\mathbf{x} \in \mathrm{Sol}(I)$ называется *системой окрестностей радиуса* k, *порожденной метрикой* $\mathcal{D}(\cdot, \cdot)$.

Алгоритм локального поиска начинает свою работу с некоторого допустимого решения. Далее на каждой итерации алгоритма происходит переход от текущего решения к новому допустимому решению в его окрестно-

сти, имеющему лучшее значение целевой функции, чем текущее решение. Процесс продолжается, пока не будет достигнут локальный оптимум. Способ выбора нового решения в окрестности текущего решения зависит от специфики конкретного алгоритма локального поиска.

1.2. Эволюционные алгоритмы

Эволюционные алгоритмы (ЭА), к числу которых относятся генетические алгоритмы, эволюционные стратегии [266], алгоритмы генетического программирования [231] и др., берут начало в работах по эволюционному моделированию Л. Фогеля с соавт. [101] и Дж. Холланда [213], где было предложено моделировать процесс биологической эволюции с целью синтеза эффективных в некотором смысле структур и создания систем искусственного интеллекта. В нашей стране А.Г. Ивахненко и Л.А. Растригиным независимо были предложены методы группового учета аргументов [55] и случайного поиска [87], где также использовались идеи эволюции. В настоящее время ЭА имеют многочисленные приложения в управлении, планировании, проектировании, распознавании образов и других областях (см., например, [84, 95, 231, 240, 266, 269]).

Характерной особенностью ЭА является имитация процесса эволюции биологической популяции, где особи соответствуют пробным точкам в пространстве решений задачи оптимизации, а *приспособленность* особей к условиям окружающей среды определяется значениями целевой функции и штрафами за нарушение ограничений задачи, если такие имеются. Как правило, процесс поиска направляется исключительно полученными значениями целевой функции в предыдущих пробных точках пространства решений. Новые решения-потомки вычисляются посредством вероятностных операторов, модифицирующих полученные ранее пробные точки.

Таким образом, при создании ЭА находит применение бионический подход, состоящий в заимствовании принципов организации систем из жи-

вой природы. В данном случае имеет место использование принципа постепенных адаптивных преобразований в пределах популяции или вида в ходе, так называемой, микроэволюции. Как показывают исследования акад. Ю.П. Алтухова и других авторов (см., [4], § 6.4), более «масштабная» межвидовая изменчивость (макроэволюция) требует скачкообразных перестроек генотипа и не может быть выведена непосредственно из постепенной внутривидовой изменчивости. В связи с недостаточной изученностью таких скачкообразных перестроек, привлечение известной теории происхождения видов для обоснования работоспособности ЭА представляется проблематичным и требуются непосредственные исследования ЭА (см. [273], § 1.3).

1.2.1. Генетические алгоритмы

Классический генетический алгоритм (КГА) предложен Дж. Холландом в [213] как алгоритм, имитирующий адаптацию популяции к окружающей среде, которая задана функцией приспособленности особей. На каждой итерации КГА с помощью рандомизированных операторов *мутации* и *кроссинговера* строится новая популяция (поколение). Численность популяции N фиксирована от начала работы алгоритма до конца. Операторы мутации и кроссинговера с некоторым упрощением моделируют процессы мутации и скрещивания в живой природе, состоящие в возникновении случайных изменений в молекулах ДНК [4].

В КГА [213] каждая особь текущей популяции выбирается в качестве родительской с вероятностью, пропорциональной ее приспособленности. Такая интерпретация приспособленности аналогична принятой в популяционной генетике [4], где под приспособленностью понимается среднее число потомков, порождаемых особью.

С 80-х годов XX века генетические алгоритмы стали активно использоваться как метод решения задач оптимизации, при этом особи представляют пробные решения задачи, а функция приспособленности в некотором

смысле характеризует их «качество». Чем выше приспособленность особи, тем больше шансов того, что она будет выбрана в качестве родительской при построении очередного потомка.

При разработке и анализе ГА важное значение имеет способ представления пробных решений в алгоритме. Во многих случаях в ГА используется тот же способ кодировки решений, что и для элементов множества Sol решаемой задачи комбинаторной оптимизации. В таком случае понятия «особь» и «пробное решение» совпадают. В противном случае термин «особь» относится только к представлению решений в ГА. Синонимом термина «особь» является «генотип». Термин «особь» используется, как правило, когда речь идет о генотипе в составе популяции.

Формально генотип ξ является строкой $(\xi_1, \xi_2, \ldots, \xi_l)$ фиксированной длины l. Элементы этой строки – символы некоторого конечного алфавита, их принято называть *генами* по аналогии с генами живых организмов, которые представляют собой участки молекулы ДНК. Во многих приложениях ГА используется алфавит $\{0, 1\}$, хотя в некоторых случаях успешно используются и недвоичные способы кодировки. Для обозначения генотипов далее, как правило, будут использоваться символы ξ, η и ζ.

Обозначим через $\mathcal{B}$ множество генотипов длины l с символами из заданного набора алфавитов $A_1, \ldots, A_l$, т. е. $\mathcal{B} = A_1 \times \ldots \times A_l$. Для применения ГА к задаче комбинаторной оптимизации необходимо определить *функцию представления решений* $\Psi : \mathcal{B} \to \{0, 1\}^n$, задающую способ представления элементов пространства решений с помощью генотипов. Далее будем предполагать, что данная функция полиномиально вычислима.

Образ $\mathbf{x} = \Psi(\xi)$ принято называть *фенотипом*, соответствующим генотипу ξ. Если кодировка решений в задаче комбинаторной оптимизации совпадает с представлением решений в ГА, то термины «генотип», «фенотип» и «пробное решение» эквивалентны, отображение $\Psi(\cdot)$ является тождественным, а вместо символов ξ, η естественно использовать $\mathbf{x}, \mathbf{y}$.

Популяция ГА представляет собой вектор $X = (\xi^1, \xi^2, \ldots, \xi^N) \in \mathcal{B}^N$, элементами которого являются генотипы. Способ нумерации особей в популяции не имеет значения. Популяция поколения t, $t = 0, 1, \ldots$, обозначается через $X^t = (\xi^{1t}, \xi^{2t}, \ldots, \xi^{Nt})$. Итерацией ГА является переход от текущей популяции X^t к следующей популяции X^{t+1}. Численность популяции N в КГА предполагается четной для удобства описания алгоритма.

В соответствии с общепринятым подходом будем предполагать, что при $\Psi(\xi) \in \mathrm{Sol}$ функция приспособленности имеет вид $\Phi(\xi) = \phi_{\mathrm{fit}}(f(\Psi(\xi)))$, где $\phi_{\mathrm{fit}} : \mathbb{R} \to \mathbb{R}$ – некоторая строго возрастающая функция, если решается задача на максимум, или строго убывающая функция в случае задачи минимизации. Если же $\Psi(\xi) \notin \mathrm{Sol}$, то функция приспособленности принимает меньшее значение, чем на любом генотипе ξ', $\Psi(\xi') \in \mathrm{Sol}$, что соответствует штрафу за нарушение ограничений. Не теряя общности, будем предполагать, что в ГА $\Phi(\xi) \geq 0$ для любого $\xi \in \mathcal{B}$. Заметим, что в случае применения ГА к задаче из класса NPO функция приспособленности указанного вида вычислима за полиномиальное время.

С использованием введенных обозначений ГА описывается следующим образом. Особи каждого нового поколения ГА строятся на основе особей текущей популяции некоторым случайным образом под действием рандомизированных операторов селекции $\mathrm{Sel} : \mathcal{B}^N \to \{1, \ldots, N\}$, мутации $\mathrm{Mut} : \mathcal{B} \to \mathcal{B}$ и кроссинговера $\mathrm{Cross} : \mathcal{B} \times \mathcal{B} \to \mathcal{B} \times \mathcal{B}$ (в некоторых генетических алгоритмах $\mathrm{Cross} : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$). Здесь и далее под *рандомизированным оператором* понимаем рандомизированную процедуру, такую что при заданном входном аргументе распределение вероятностей на ее выходе не зависит от предшествующей работы алгоритма. Будем предполагать, что вычисления рандомизированной процедуры моделируются вероятностной машиной Тьюринга [59]. Приведем схему *генетического алгоритма с полной заменой популяции*. Этой схеме соответствует, в част-

ности, КГА [197, 213].

Алгоритм 1.1. ГА с полной заменой популяции

1. Положить $t := 0$.

2. Для k от 1 до N выполнять:

2.1. Построить случайным образом особь $\xi^{k,0}$.

Итерация t.

3. Для k от 1 до $N/2$ выполнять шаги 3.1–3.3:

3.1. Селекция: выбрать особи $\xi := \xi^{\mathrm{Sel}(X^t),t}$, $\eta := \xi^{\mathrm{Sel}(X^t),t}$.

3.2. Кроссинговер: построить $(\xi', \eta') := \mathrm{Cross}(\xi, \eta)$.

3.3. Мутация: положить $\xi^{2k-1,t+1} := \mathrm{Mut}(\xi')$, $\xi^{2k,t+1} := \mathrm{Mut}(\eta')$.

4. Положить $t := t + 1$.

5. Если условие остановки не выполнено, то идти на шаг 3.

6. Результатом работы является особь с наибольшей приспособленностью среди найденных за все итерации.

Поясним приведенную схему. На шаге 2 формируется начальная популяция X^0, элементы которой генерируются некоторой детерминированной или рандомизированной процедурой, например, равновероятно на множестве $\mathcal{B}$.

Оператор селекции имеет то же значение, что и естественный отбор в природе. Действие этого оператора состоит в выборе номера родительской особи для построения очередного потомка. Оператор *пропорциональной селекции*, предложенный в КГА [213], основан на выборе родительских решений с вероятностями, пропорциональными приспособленности особей в текущей популяции. Генотип $\xi^{i,t}$ с номером i, $i = 1, \ldots, N$, из популяции X^t оказывается родительской особью при формировании очередного генотипа $\xi^{k,t+1}$ популяции X^{t+1} с вероятностью $\Phi(\xi^{i,t}) / \sum_{j=1}^{N} \Phi(\xi^{j,t})$.

Наряду с пропорциональной селекцией будем рассматривать другой широко известный оператор *турнирной селекции* [199]. При действии дан-

ного оператора из популяции равновероятно (с возвращением) выбираются s особей и в качестве родителя используется лучшая по приспособленности среди выбранных. Параметр s называется размером турнира.

Размер популяции N и размер турнира s, вообще говоря, зависят от индивидуальной задачи I, и их выбор может существенно влиять на скорость сходимости популяции к решениям приемлемого качества.

Условие остановки (шаг 5) может быть сформулировано по-разному. Например, по достижении заданного значения целевой функции, по достижении заданного числа итераций или по прошествии заданного числа итераций без улучшения значения целевой функции рекордного решения. Под рекордным решением понимается лучшее найденное решение (по целевой функции) с начала выполнения алгоритма до текущего момента.

Кроме описанной выше схемы ГА известны и другие варианты этого алгоритма (см., например, [84,240,269,273,277]). Во многих ГА используется *стационарная стратегия управления популяцией* [269]:

Алгоритм 1.2. ГА со стационарной стратегией управления популяцией

1. Положить $t := 0$.

2. Для k от 1 до N выполнять:

2.1. Построить случайным образом особь $\xi^{k,0}$.

Итерация t.

3. Селекция: выбрать особи $\xi := \xi^{\mathrm{Sel}(X^t),t}$, $\eta := \xi^{\mathrm{Sel}(X^t),t}$.

4. Кроссинговер: построить $(\xi', \eta') := \mathrm{Cross}(\xi, \eta)$.

5. Мутация: построить $\xi'' := \mathrm{Mut}(\xi')$, $\eta'' := \mathrm{Mut}(\eta')$.

6. Удалить из популяции X^t две особи с наименьшей приспособленностью и поместить на их место генотипы ξ'' и η''.

7. Положить $X^{t+1} := X^t$, $t := t + 1$.

8. Если условие остановки не выполнено, то идти на шаг 3.

9. Результатом работы является особь с наибольшей приспособленностью

среди найденных за все итерации.

На каждой итерации ГА с такой стратегией управления популяцией изменяются всего две особи. Если же используется оператор кроссинговера $\mathrm{Cross} : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$, вычисляющий только один генотип, то η' и η'' на шагах 4,5 и 6 не обрабатываются и в популяции обновляется только одна особь. В некоторых других вариантах ГА со стационарной стратегией управления популяцией выбор особей на шаге 6 осуществляется случайным образом, причем особи с меньшим значением приспособленности имеют большую вероятность удаления (см., например, [269] и п. 5.2.3 данной работы).

Операторы мутации и кроссинговера классического генетического алгоритма. Наибольший интерес представляют несколько вариантов мутации и кроссинговера, широко используемых в ГА и моделирующих процессы рекомбинации и мутации в живой природе. Рассмотрим операторы мутации Mut^* и кроссинговера Cross^*, предложенные для классического ГА с бинарным пространством генотипов $\mathcal{B} = \{0,1\}^l$. Приведем описание этих операторов для случая произвольных алфавитов $A_1, \ldots, A_l$ в связи с тем, что эти операторы используются не только в КГА, но и в других ГА, и, в частности, при $\mathcal{B} \neq \{0,1\}^l$. Результат кроссинговера [197] $(\xi', \eta') = \mathrm{Cross}^*(\xi, \eta)$ с вероятностью P_c имеет вид

$$\xi' = (\xi_1, \xi_2, ..., \xi_\chi, \eta_{\chi+1}, ..., \eta_l),$$

$$\eta' = (\eta_1, \eta_2, ..., \eta_\chi, \xi_{\chi+1}, ..., \xi_l),$$

где координата χ выбрана равновероятно от 1 до $l-1$. С вероятностью $1 - P_\mathrm{c}$ родительские особи копируются без изменений, т. е. $\xi' = \xi,\ \eta' = \eta$. Данный оператор Cross^* принято называть *одноточечным кроссинговером.*

При действии Mut^* вычисляется особь $\xi' = \mathrm{Mut}^*(\xi)$, где каждому гену $\xi'_i,\ i = 1, \ldots, l$, независимо от других, с заданной вероятностью P_m присваивается значение, отличное от ξ_i, а с вероятностью $1 - P_\mathrm{m}$ сохраняется

значение ξ_i. В случае $|A_i| > 2$ выбор нового значения для ξ_i осуществляется некоторым случайным образом.

Степень воздействия кроссинговера и мутации в КГА регулируется параметрами P_{c} и P_{m}, которые, вообще говоря, могут зависеть от I. Увеличение вероятности мутации до 0.5 превращает ГА в простой случайный перебор. Уменьшение P_{m} до нуля приводит к малому разнообразию в популяции и может вызвать «зацикливание» ГА, когда на каждой итерации генерируются лишь ранее встречавшиеся особи.

Другие операторы мутации и кроссинговера. Известно множество вариантов генетического алгоритма, имеющих разные операторы селекции, мутации, кроссинговера и несколько различающихся в своих схемах [269]. Рассмотрим некоторые часто используемые мутации и кроссинговера.

Как в природе, так и во многих известных генетических алгоритмах каждый из генов подвергается мутации независимо от других, однако вероятность мутации в разных генах может быть различной.

При малой вероятности мутации в КГА возможность одновременного изменения более одного гена особенно мала. В таких ситуациях естественным образом возникает оператор точечной мутации [277], где с вероятностью P' происходит равновероятный выбор номера мутируемого гена $i \in \{1, \ldots, l\}$, и только в этом гене генотип потомка приобретает отличие от родительского. С вероятностью $1 - P'$ генотип остается без изменений.

В некоторых вариантах кроссинговера результатом является один генотип, однако наиболее распространены операторы с двумя выходными генотипами. Их общей чертой является так называемое свойство *передачи генов*: значение для каждого гена потомка выбирается из значений соответствующих генов одного или другого родителей [**?**].

Для описания операторов со свойством передачи генов уместно использовать *маску кроссинговера*. Под этим термином понимают вспомо-

гательную последовательность $\mathbf{m} = (m_1, \ldots, m_l) \in \{0,1\}^l$, по которой с вероятностью P_c строятся генотипы потомков:

$$\xi'_i := \begin{cases} \xi_i, & \text{если } m_i = 1 \\ \eta_i, & \text{иначе,} \end{cases} \quad ; \quad \eta'_i := \begin{cases} \eta_i, & \text{если } m_i = 1 \\ \xi_i, & \text{иначе,} \end{cases}$$

для $i = 1, \ldots, l$; в противном случае (с вероятностью $1 - P_\text{c}$) $\xi' := \xi$, $\eta' := \eta$. Рассмотрим два примера использования маски кроссинговера.

Равномерный кроссинговер соответствует равновероятному выбору маски кроссинговера из множества $\{0,1\}^l$. В таком случае каждый ген копируется в генотип потомка из соответствующей позиции генотипа одного или другого родителя с равными вероятностями.

k-точечный кроссинговер представляет собой обобщение одноточечного. Выбирается k различных координат скрещивания $0 < \chi_1 < \chi_2 < \ldots < \chi_k < l$ равновероятно среди всевозможных таких наборов. Пусть $\chi_0 = 0$, тогда маска кроссинговера имеет вид:

$$m_i = \begin{cases} 1, & \text{если } \max\{j : \chi_j < i\} \text{ – четное число} \\ 0, & \text{иначе} \end{cases}, \qquad i = 1, \ldots, l.$$

Каждый из описанных операторов кроссинговера может быть реализован в варианте с одним генотипом потомка при исключении второго генотипа.

1.2.2. Общие схемы эволюционных алгоритмов

Приведем описание общего вида операторов ЭА и его общую схему. Определения фенотипа, функции представления решений и популяции, введенные выше для ГА, сохраняются без изменений. Функция приспособленности генотипа $\Phi(\xi)$ в случае ЭА может иметь более общий вид, чем в случае ГА. Известны эволюционные алгоритмы как для задач с одним критерием оптимизации, так и для многокритериальных задач, поэтому допускается и многокритериальная вектор-функция приспособленности.

Обозначим для любой популяции $X = (\xi^1, \ldots, \xi^N)$ множество представленных в ней генотипов через $\hat{X}$, т. е. $\hat{X} = \bigcup_{j=1}^{N} \{\xi^j\}$. Каждая новая популяция X^{t+1} строится на основании текущей популяции X^t с помощью следующих операторов (см., например, [179, 277]).

Оператором *отбора* $\mathrm{Sel} : \mathcal{B}^N \to \mathcal{B}^{N'}$ извлекаются N' копий родительских генотипов из текущей популяции. Пусть промежуточная популяция $X' = \mathrm{Sel}(X^t)$, тогда $\hat{X}' \subseteq \hat{X}^t$.

Оператор *воспроизведения* $\mathrm{Rep} : \mathcal{B}^{N'} \to \mathcal{B}^{N''}$ вносит некоторые случайные изменения в родительские генотипы, создавая при этом N'' генотипов потомков, составляющих промежуточную популяцию X''.

Оператором *выживания* $\mathrm{Surv} : \mathcal{B}^N \times \mathcal{B}^{N''} \to \mathcal{B}^N$ определяются те генотипы из X^t и X'', которые составят X^{t+1}, т. е. $\hat{\mathrm{Surv}}(X^t, X'') \subseteq \hat{X}^t \cup \hat{X}''$.

Будем допускать, что распределение на выходе операторов Sel, Rep и Surv может также зависеть от номера итерации, но для компактности обозначений символ t не будем указывать в числе аргументов. Численности популяций N, N' и N'' могут изменяться в процессе работы ЭА.

Пусть оператор Init получает на вход только исходные данные задачи и вычисляет начальную популяцию $X^0 \in \mathcal{B}^N$.

Работа ЭА начинается с популяции $X^0 = \mathrm{Init}$ и продолжается итерациями следующей композиции операторов:

$$X^{t+1} = \mathrm{Surv}(\mathrm{Rep}(\mathrm{Sel}(X^t)), X^t), \quad t \in \mathbb{Z}_+.$$

Здесь и далее $\mathbb{Z}_+$ обозначает множество неотрицательных целых чисел. Вычисления оканчиваются при выполнении некоторого правила остановки (например, по числу итераций или «качеству» полученных решений). Результатом работы ЭА является одно или несколько лучших по приспособленности решений, полученных за все время работы алгоритма.

Заметим, что общий вид рандомизированных операторов допускает некоторую зависимость распределения вероятностей на выходе таких опе-

раторов от номера итерации ЭА – достаточно, например, в состав генотипа ввести набор генов, содержащих номер текущей итерации. Выбор операторов ЭА осуществляется для рассматриваемого конкретного класса задач с учетом специфики этих задач. Экспериментальная практика, а также некоторые теоретические работы (см., например, [158,159,245]) показывают, что выбор адекватных операторов отбора, воспроизведения и выживания играет ключевую роль в обеспечении работоспособности ЭА.

Генетические алгоритмы входят в класс ЭА. Сюда же относятся многие алгоритмы генетического программирования [231], эволюционные стратегии [132, 266], алгоритмы поиска с запретами [24, 194] и имитации отжига [227], алгоритм «иди за победителями» [109], СПА [76] и др.

О применимости ЭА к индивидуальной задаче комбинаторной оптимизации естественно судить по математическому ожиданию времени первого достижения оптимального решения или достаточно точного приближенного решения. Эти средние значения могут быть оценены снизу и сверху функциями от длины записи исходных данных задачи, что позволяет делать вывод об эффективности ЭА на задаче комбинаторной оптимизации в целом. Существование рандомизированных алгоритмов с полиномиальной средней трудоемкостью получения решения для какой-либо NP-трудной задачи противоречило бы известной гипотезе о неравенстве классов $\mathrm{NP} \neq \mathrm{RP}$ (что следует, например, из леммы 11 [135] ввиду неравенства Маркова). В связи с этим существование ЭА, обнаруживающего оптимальное решение NP-трудной задачи в среднем за время, полиномиально ограниченное сверху, также представляется маловероятным.

Теоретически ЭА до сих пор изучены недостаточно, несмотря на активные исследования. Результаты в этой области либо дают оценки быстродействия ЭА на специальных классах задач (см., например, [89, 288]), либо имеют достаточно общий характер, но их применение к конкретной задаче оптимизации с целью получения нетривиальных оценок быстродей-

ствия алгоритма весьма проблематично [276, 289, 290]. Обзоры литературы по теории ЭА могут быть найдены в [132, 273]. Особый интерес представляет сравнение ЭА с классическими методами дискретной оптимизации. В последние годы новые результаты получены при анализе ЭА для таких базовых задач комбинаторной оптимизации, как задача об остовном дереве минимального веса [249], задача о наибольшем паросочетании [193] и задача о минимальном разрезе в графе [247]. Известны полиномиальные в среднем ЭА для поиска приближенных решений NP-трудных задач, например, для ЗНП [186]. В [267] предложена общая схема ЭА с полиномиальным в среднем временем поиска решения для задач оптимизации на матроидах.

Для ряда задач комбинаторной оптимизации известны ЭА, имеющие среднее время получения оптимума, близкое к трудоемкости алгоритмов динамического программирования (ДП). В частности, такие алгоритмы предложены для задачи о кратчайшем пути в графе [279], задачи коммивояжера [286], задачи об одномерном булевом рюкзаке [35] и целого класса задач [155], включающего задачи многокритериальной оптимизации.

Эволюционные стратегии (μ, λ)-ES, $(\mu + \lambda)$-ES. Один из первых вариантов эволюционной стратегии (1+1)-ES был предложен Л.А. Растригиным и представляет собой рандомизированный вариант локального поиска (см. алгоритм локального поиска с пересчетом при неудачном шаге в [87], гл. 2). И. Реченберг [266] сформулировал более общие вычислительные схемы эволюционных стратегий, которые приводятся ниже.

Прежде чем дать описание эволюционных стратегий, определим вспомогательный оператор s_μ, который из данной на вход популяции генотипов (численностью не менее μ) выбирает μ особей с наибольшей приспособленностью и возвращает популяцию из этих особей в качестве результата.

Алгоритм 1.3. Общая схема (μ, λ)-ES и $(\mu + \lambda)$-ES

1. Построить популяцию $X^{(0)} = (\xi^{1,0}, \ldots, \xi^{\mu,0})$ случайным образом.

2. Для t от 0 до $t_{max} - 1$ выполнять:

2.1 Для i от 1 до λ выполнять 2.1.1, 2.1.2:

2.1.1 Выбрать u_i с равновероятно из $\{1, 2, \ldots, \mu\}$.

2.1.2 Положить $\eta^i := \mathrm{Mut}(\xi^{u_i,t})$.

2.2 $X^{t+1} := \begin{cases} s_\mu(\eta^1, \ldots, \eta^\lambda) \text{ в алгоритме } (\mu, \lambda)\text{-ES} \\ s_\mu(\xi^{1,t}, \ldots, \xi^{\mu,t}, \eta^1, \ldots, \eta^\lambda) \text{ в алгоритме } (\mu + \lambda)\text{-ES.} \end{cases}$

2.3 $t := t + 1$.

3. Результат – особь с наибольшей приспособленностью за все итерации.

Многокритериальный ЭА. Рассмотрим задачу многокритериальной оптимизации. Пусть на конечном множестве (пространстве поиска) $\mathcal{S}$ задана векторнозначная целевая функция $g: \mathcal{S} \to \mathbb{R}^m$, каждая компонента которой является критерием максимизации. В таком случае $g(\mathcal{S}) \subseteq \mathbb{R}^m$ – *пространство критериев*. Введем частичный порядок $\succeq$, обозначающий доминирование по Парето: $(y'_1, \ldots, y'_m) \succeq (y_1, \ldots, y_m)$ тогда и только тогда, когда $y'_j \leq y_j$ для всех $j = 1 \ldots, m$ ($y' \succ y$, если $y' \succeq y$ и $y \not\succeq y'$). *Фронтом Парето* называется множество максимальных элементов по отношению $\succeq$ во множестве $g(\mathcal{S})$. Решение задачи состоит в отыскании полного множества альтернатив (ПМА), т. е. минимального по включению подмножества в $\mathcal{S}$, которое отображается во фронт Парето под действием g.

Для решения задач многокритериальной оптимизации предложено большое число ЭА, основанных на вычислительных схемах известных эволюционных алгоритмов, таких как генетические алгоритмы, эволюционные стратегии и алгоритмы генетического программирования (см., например, [36, 90, 248, 279]). По-видимому, первый ЭА с вычислительной схемой, специально предназначенной для решения задач многокритериальной оптимизации, может быть найден в работе [257] М. Пешеля и К. Риделя. Приведем общую схему многокритериального ЭА, основанную на алгоритме из [257]. Здесь популяция X не может содержать одинакове особи, поэтому

вместо X в алгоритме будет, как правило, удобнее использовать множество генотипов $\hat{X}$. Для простоты обозначений полагаем, что пространство генотипов совпадает с пространством поиска $\mathcal{S}$.

Алгоритм 1.4. Многокритериальный эволюционный алгоритм

1. $Y' := \mathrm{Init}$.
2. $\hat{X} := \emptyset$.
3. Для всех $\xi \in \hat{Y}'$:
4. Если $\nexists \xi' \in \hat{X}$: $g(\xi) \preceq g(\xi')$, то
5. $\hat{X} := (\hat{X} \setminus \{\xi' \in \hat{X} \mid g(\xi') \prec g(\xi)\}) \cup \{\xi\}$.
6. Конец если.
7. Конец цикла.
8. Пока не выполнен критерий остановки :
9. $Y' := \mathrm{Rep}(X)$.
10. Для всех $\xi \in \hat{Y}'$:
11. Если $\nexists \xi' \in \hat{X}$: $g(\xi) \preceq g(\xi')$, то
12. $\hat{X} := (\hat{X} \setminus \{\xi' \in \hat{X} \mid g(\xi') \prec g(\xi)\}) \cup \{\xi\}$.
13. Конец если.
14. Конец цикла.
15. Конец цикла.
16. Результат: $\hat{X}$.

Здесь Init случайным образом генерирует начальную популяцию. Рандомизированный оператор воспроизведения имеет вид $\mathrm{Rep} : \mathcal{S}^* \to \mathcal{S}^{N''}$, где $\mathcal{S}^*$ – множество всевозможных популяций с особями из $\mathcal{S}$; N'' – фиксированная численность промежуточной популяции. Алгоритмы, соответствующие данной схеме, будем называть *многокритериальными ЭА*. Примерами таких алгоритмов при $N'' = 1$ являются алгоритмы SEMO и FEMO из [234], GSEMO из [248] и др. В [278] для алгоритмов подобного типа получены условия сходимости $g(\hat{X})$ к фронту Парето с вероятностью 1.

1.2.3. Гибридные эволюционные алгоритмы

На практике приведенные выше общие схемы эволюционных алгоритмов зачастую используются в комбинации с известными алгоритмами, учитывающими специфику решаемой задачи. Работу таких гибридных алгоритмов сложнее интерпретировать как моделирование эволюции биологической популяции, чем в случае непосредственного использования базовой схемы ЭА. Однако, как отмечено Л.А. Растригиным [265], «... стремление к моделям биологической эволюции и генетических механизмов не должно быть чрезмерным, так как они созданы природой (и/или Творцом) для развития биологических существ. Переносить их без поправок на развитие технических объектов было бы серьезной методологической ошибкой».

Наиболее часто в общую схему ЭА включаются методы локальной оптимизации, жадные алгоритмы корректировки решений и декодирующие эвристики [5, 84]. Также специфика задачи в ЭА может быть использована при решении впомогательных задач математического программирования в операторах кроссинговера и при построении начальной популяции. Большое практическое значение имеют комбинации ЭА с такими точными методами, как метод ветвей и границ, перебор L-классов и декомпозиция Бендерса (см., например, [9, 98, 177]). При этом работа ЭА и точного алгоритма может осуществляться параллельно, с обменом наилучшими найденными решениями [177]. Вычислительные эксперименты с гибридными алгоритмами описанного типа показали, что применение ЭА в комбинации с точными методами позволяет существенно сократить время поиска оптимального решения и полное время работы алгоритма.

Использование методов целочисленного линейного программирования рассматривается в главах 3 и 5. Ниже описываются основные принципы работы эвристических процедур в составе ЭА.

Локальная оптимизация. Рассмотрим метод, состоящий в применении процедуры локальной оптимизации к каждому новому генотипу, полученному в ЭА. Мотивацией для использования локального поиска в ЭА является тот факт, что множество локальных оптимумов содержит, в частности, и глобальный оптимум. Поэтому естественно предполагать, что организация поиска на множестве локальных оптимумов или в окрестности этого множества увеличит шансы получить глобальный оптимум. В наибольшей степени такой подход пригоден для тех случаев, когда вычисление значения $f(\mathbf{y})$, $\mathbf{y} \in \mathcal{N}_I(\mathbf{x})$ значительно упрощается при известном значении $f(\mathbf{x})$. Однако этот подход показывает хорошие экспериментальные результаты и в других ситуациях (см., например, [161, 163]).

Улучшенное локальным поиском решение кодируется как генотип и добавляется в популяцию ЭА, либо в популяции сохраняется исходный генотип, а значение целевой функции в полученном локальном оптимуме служит для вычисления его приспособленности. В первом случае локальная оптимизация может рассматриваться как компонент оператора мутации (в англоязычной литературе такие ЭА получили наименование *memetic algorithms* [243]), а во втором – как вычисление отображения $\Psi(\xi)$, декодирующего генотип.

При использовании первого подхода возникает вопрос о мощности множества всех генотипов, получаемых в результате локальной оптимизации, так как мощность этого множества характеризует сложность решаемой задачи для ЭА. В [180, 181, 272] нами предложены статистические методы оценивания числа локальных оптимумов задачи комбинаторной оптимизации на основе результатов многократного выполнения локального поиска. Предложенные методы опробованы на ряде задач комбинаторной оптимизации. Среди них задачи с целевыми функциями, построенными случайным образом по модели nk-ландшафтов С. Кауффмана [225], задачи отыскания двоичной последовательности с минимальной автокорреля-

цией [198], задачи о вершинном покрытии и др.

Жадные алгоритмы корректировки решений. Данный метод подобен локальной оптимизации решений, но отличается тем, что к полученному пробному решению вместо локальной оптимизации применяют некоторый жадный алгоритм. Как правило, для этого требуется представить задачу в виде задачи отыскания набора или размещения элементов J из некоторого заданного множества U.

Для корректировки генотипа ξ, полученного в результате действия операторов мутации и кроссинговера, к набору элементов $J(\xi)$, соответствующих ξ, применяется жадный алгоритм. В некоторых случаях в жадном алгоритме целесообразно использовать только элементы из $J(\xi)$ (см., например, ГА для задачи о покрытии в разделе 5.1). В других случаях в процессе работы жадного алгоритма в множество $J(\xi)$ добавляются элементы из $U \backslash J(\xi)$ (см., например, [127]).

Декодирующие эвристики. Для многих задач дискретной оптимизации имеется возможность декомпозиции исходной задачи на ряд более простых подзадач, причем получающиеся подзадачи оказываются однотипными и могут быть эффективно решены точно или приближенно. К числу задач рассматриваемого класса относятся такие, где требуется найти оптимальный набор или размещение элементов из некоторого заданного множества, задачи составления расписаний, маршрутизации перевозок и др. Как правило, решение подзадач может быть осуществлено с помощью одного или нескольких альтернативных жадных эвристических правил, в результате чего к имеющемуся (возможно, недопустимому) частичному решению добавляется один или более новых элементов.

Повышение точности ЭА может быть достигнуто за счет подбора последовательности эвристик, применяемых для решения подзадач, или за

счет выбора порядка обработки элементов решения. Для поиска наиболее подходящего упорядочивания может быть использован генетический алгоритм, где генотип содержит информацию о последовательности применения эвристик или обработки элементов решения. Данный подход успешно применяется к задачам составления расписаний, двумерной упаковки, распределения заданий по локальной вычислительной сети [84], к задачам управления поставками продукции (см. п. 5.2.3) и др.

2. Исследование эволюционных алгоритмов с позиций локальной оптимизации

Настоящая глава имеет следующую структуру.

В разделе 2.2 получены оценки числа приближенных решений достаточно высокого качества на заданной итерации генетического алгоритма с турнирной селекцией и установлен класс одноэкстремальных функций, на котором полученные оценки оказываются достижимы.

В разделах 2.1 и 2.3 проведено сравнение методов локального поиска и эволюционных алгоритмов. В разделе 2.1 показано, что на классе задач NP-оптимизаци с гарантированными локальными оптимумами генетический алгоритм с турнирной селекцией достигает локальный оптимум в среднем за полиномиально ограниченное время. Далее, в разделе 2.3, найдены необходимые и достаточные условия, при которых рандомизированный вариант локального поиска, (1+1)-ES, является наилучшим методом с точки зрения вероятности отыскания оптимума на каждой итерации в классе эволюционных алгоритмов с заданным оператором мутации.

2.1. Генетический алгоритм как метод локального поиска

Настоящий раздел посвящен изучению достаточных условий, при которых генетический алгоритм с полной заменой популяции и турнирной селекцией впервые посещает локальный оптимум в среднем за полиномиальное от длины исходных данных время.

Мотивацией исследования служит тот факт, что ГА зачастую относят к классу методов локального поиска (см., например, [71, 107, 268]), поэтому представляет интерес детальное изучение случаев, когда работоспособ-

ность ГА объясняется сходством его поведения с локальным поиском.

Для простоты обозначений здесь предполагается двоичное представление решений, совпадающее с кодировкой решений задачи комбинаторной оптимизации, а «генотип» – то же, что элемент пространства решений $\{0,1\}^{n(I)}$. В связи с этим для обозначения генотипов, как правило, будут использоваться символы $\mathbf{x}$ или $\mathbf{y}$. Исследуется ГА с полной заменой популяции и турнирной селекцией (см. алгоритм 1.1) в предположении, что всякий раз при выполнении условия остановки осуществляется переход на шаг 1 и вычислительный процесс продолжается неограниченно.

Будем предполагать, что в результате кроссинговера с вероятностью не менее некоторой константы ε, $0 < \varepsilon \leq 1$, образуются особи $(\mathbf{x}', \mathbf{y}') = \mathrm{Cross}(\mathbf{x}, \mathbf{y})$, хотя бы одна из которых не уступает по приспособленности родительским особям $\mathbf{x}, \mathbf{y} \in B$, т. е.

$$\mathbf{P}\{\max\{\Phi(\mathbf{x}'), \Phi(\mathbf{y}')\} \geq \max\{\Phi(\mathbf{x}), \Phi(\mathbf{y})\}\} \geq \varepsilon \tag{2.1}$$

при любых $\mathbf{x}, \mathbf{y} \in B$, причем константа ε не зависит от I.

Для одноточечного кроссинговера условие (2.1) выполняется с $\varepsilon = 1 - P_{\mathrm{c}}$, если $P_{\mathrm{c}} < 1$ – константа, не зависящая от I. Условие (2.1) выполняется с $\varepsilon = 1$, если один из двух потомков – решение задачи оптимальной рекомбинации родительских решений (см. главу 3).

Далее ГА, соответствующий приведенному описанию, будет обозначаться через GA.

Раздел имеет следующую структуру: в п. 2.1.1 получена оценка среднего времени первого достижения локального оптимума; в п. 2.1.2 с использованием найденной оценки исследуется среднее время получения оптимума в двух специальных семействах задач; в п. 2.1.3 оценка из раздела 2.1.1 применяется к классу задач с гарантированными локальными оптимумами; п. 2.1.4 содержит заключительные замечания.

2.1.1. Среднее время достижения локального оптимума

Пусть имеется задача максимизации $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f_I) \in \mathrm{NPO}$ и выбрана некоторая система окрестностей $\{\mathcal{N}(\mathbf{x}) \mid \mathbf{x} \in \mathrm{Sol}(I)\}$. Обозначим через h число всех неоптимальных значений целевой функции f, т. е. $h = |\{f(\mathbf{x}) : \mathbf{x} \in \mathrm{Sol}\}| - 1$. Тогда начиная с любого решения локальный поиск достигает локального оптимума не более чем за h улучшающих целевую функцию итераций. Пусть L обозначает минимальную вероятность достижения решения в пределах окрестности:

$$L = \min_{\mathbf{x}\in\mathrm{Sol},\ \mathbf{x}'\in\mathcal{N}(\mathbf{x})} \mathbf{P}\{\mathrm{Mut}(\mathbf{x}) = \mathbf{x}'\}.$$

Чем выше величина L, тем больше согласованность оператора мутации с системой окрестностей. Численность популяции N, размер турнира s и величину L будем рассматривать как функции от исходных данных задачи I.

Пусть e – число Эйлера. Докажем техническую лемму.

Лемма 2.1. *Если X^0 содержит допустимое решение, $s \geq rN$, $r > 0$, $h > 1$, $L > 0$,*

$$N \geq \frac{2(1 + \ln h)}{L\varepsilon(1 - 1/e^{2r})}, \tag{2.2}$$

и в GA остановка и новая инициализация выполняется при условии $t = h + 1$, то

1) *GA посещает локальный оптимум к итерации h с вероятностью не менее $1/e$ и*

2) *локальный оптимум достигается не позднее, чем за eh итераций GA в среднем.*

Доказательство. Заметим, что в начальной популяции особь с наибольшей приспособленностью является допустимым решением, так как приспособленность особей, отвечающих недопустимым решениям, всегда меньше приспособленности решений из допустимой области. Пусть событие E_k^{t+1}, $k = 1, \ldots, N/2$, состоит в выполнении следующих трех условий:

1. из популяции X^t при построении k-той пары потомков следующего поколения выбирается решение $\mathbf{x}_*^t$ наибольшей приспособленности;

2. при построении k-той пары потомков посредством кроссинговера один из них имеет приспособленность не менее $\Phi(\mathbf{x}_*^t)$ (пусть для определенности это $\mathbf{x}'$);

3. оператор мутации, примененный к $\mathbf{x}'$, осуществляет переход в наилучшее по приспособленности решение в окрестности $\mathcal{N}(\mathbf{x}')$, т. е. $\Phi(\mathrm{Mut}(\mathbf{x}')) = \max_{\mathbf{y}\in\mathcal{N}(\mathbf{x}')} \Phi(\mathbf{y})$.

Обозначим через p вероятность наступления хотя бы одного из событий $E_k^{t+1},\ k = 1, \ldots, N/2$, при известной популяции X^t. Найдем оценку $\lambda \le p$, не зависящую от выбора X^t. Согласно схеме GA, $\mathbf{P}\{E_1^{t+1}\} = \ldots = \mathbf{P}\{E_{N/2}^{t+1}\}$. Обозначим эту вероятность через q. Ввиду независимости событий $E_k^{t+1},\ k = 1, \ldots, N/2$, при фиксированной X^t имеем $p \ge 1-(1-q)^{N/2} \ge 1 - e^{-qN/2}$. Оценим снизу вероятность q:

$$q \ge L\varepsilon \left(1 - \left(1 - \frac{1}{N}\right)^{2s}\right).$$

Однако $(1 - 1/N)^{2s} \le (1 - 1/N)^{2rN} \le 1/e^{2r}$, поэтому

$$q \ge L\varepsilon \left(1 - \frac{1}{e^{2r}}\right) = Lc, \tag{2.3}$$

где $c = \varepsilon\left(1 - \frac{1}{e^{2r}}\right)$. В дальнейшем мы воспользуемся тем, что из (2.2) и (2.3) вытекает

$$N \ge \frac{2}{L\varepsilon\,(1 - 1/e^{2r})} \ge 2/q. \tag{2.4}$$

Для оценки снизу вероятности p сначала заметим, что при любом $z \in [0, 1]$

$$1 - \frac{z}{e} \ge e^{-z}. \tag{2.5}$$

Положим $z = e^{-qN/2+1}$. Тогда ввиду неравенства (2.4) имеем $z \le 1$ и, следовательно,

$$p \ge \exp\left\{-e^{1-qN/2}\right\} \ge \exp\left\{-e^{1-LcN/2}\right\}. \tag{2.6}$$

От анализа потомков фиксированной популяции X^t перейдем к случайной последовательности популяций $X^0, X^1, \ldots$. Заметим, что λ^h является оценкой снизу для вероятности достичь локальный оптимум за серию из не более h итераций, улучшающих значение рекорда целевой функции. Действительно, пусть $A_t = E_1^t + \ldots + E_{N/2}^t,\ t = 1, 2, \ldots$. Тогда

$$\mathbf{P}\{A_1 \& \ldots \& A_h\} = \mathbf{P}\{A_1\} \prod_{t=1}^{h-1} \mathbf{P}\{A_{t+1} | A_1 \& \ldots \& A_t\} \geq \lambda^h. \qquad (2.7)$$

Итак, положим $\lambda = \exp\left\{-e^{1-LcN/2}\right\}$. Снова воспользовавшись условием (2.2), получаем оценку снизу для вероятности достичь локальный оптимум за серию из не более h улучшающих рекорд итераций:

$$\lambda^h = \exp\left\{-he^{1-LcN/2}\right\} \geq \exp\left\{-he^{-\ln h}\right\} = 1/e.$$

Первая часть утверждения леммы доказана.

Для оценки среднего времени получения локального оптимума рассмотрим последовательность серий по h итераций в каждой. Пусть событием $D_i,\ i = 1, 2, \ldots,$ является отсутствие локального оптимума в популяции GA в i-той серии. При выполнении условий леммы вероятность каждого события $D_i,\ i = 1, 2, \ldots,$ не превышает $\mu = 1 - 1/e$ при любой предыстории работы алгоритма. По аналогии с (2.7) заключаем: $\mathbf{P}\{D_1 \& \ldots \& D_k\} \leq \mu^k$. Таким образом, если через η обозначить случайную величину, равную номеру первой серии, на которой локальный оптимум будет получен, то, пользуясь свойствами математического ожидания (см., например, [25]), получаем

$$E[\eta] = \sum_{i=0}^{\infty} \mathbf{P}\{\eta > i\} = 1 + \sum_{i=1}^{\infty} \mathbf{P}\{D_1 \& \ldots \& D_i\} \leq 1 + \sum_{i=1}^{\infty} \mu^i = e.$$

Следовательно, локальный оптимум достигается не позднее, чем за eh итераций GA в среднем. $\square$

Пусть $\lceil \cdot \rceil$ обозначает округление вверх. Тогда в условиях леммы, при

$$N = 2 \left\lceil \frac{1 + \ln h}{L\varepsilon(1 - 1/e^{2r})} \right\rceil, \quad s = \lceil rN \rceil, \qquad (2.8)$$

обеспечено получение локального оптимума в GA за $O(h)$ итераций в среднем. Трудоемкости операторов Mut и Cross полиномиально ограничены, при $\Pi \in \mathrm{NPO}$ функция приспособленности полиномиально вычислима, а процедура турнирной селекции требует времени $O(s) = O(N)$. Следовательно, имеет место

Теорема 2.1. *Если задача* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f_I) \in \mathrm{NPO}$ *полиномиально ограничена, функция* $1/L(I)$ *полиномиально ограничена, а популяция* X^0 *содержит допустимое решение, то при соответствующем выборе параметров GA локальный оптимум впервые достигается в среднем за полиномиально ограниченное время.*

Если мощность $|\mathcal{N}(\mathbf{x})|$ при любом $\mathbf{x} \in \mathrm{Sol}$ ограничена полиномом от $|x|$, то отображение $\mathcal{N}$ называют *полиномиально ограниченным* [275]. Для такого $\mathcal{N}$ существует оператор мутации $\mathrm{Mut}(\mathbf{x})$, вычислимый за полиномиально ограниченное время и осуществляющий равновероятный выбор особей-потомков из множества $\mathcal{N}(\mathbf{x})$ при заданном $\mathbf{x}$. Тогда $1/L$ также ограничена сверху некоторым полиномом от $|I|$. Таким образом, теорема 2.1 применима ко многим известным системам окрестностей для задач комбинаторной оптимизации.

Пусть $\delta(\mathbf{x}, \mathbf{y})$ обозначает расстояние Хэмминга между двоичными строками $\mathbf{x}$ и $\mathbf{y}$.

Определение 2.1. [116] *Система окрестностей* $\{\mathcal{N}(\mathbf{x}) \mid \mathbf{x} \in \mathrm{Sol}(I)\}$ *называется* k-ограниченной, *если для любых* $\mathbf{x} \in \mathrm{Sol}$ *и* $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ *выполнено* $\delta(\mathbf{x}, \mathbf{y}) \leq k$, *где* k *– константа.*

Оператор мутации Mut^* классического ГА [213] с вероятностью $P_\mathrm{m}^{\delta(\mathbf{x},\mathbf{y})}(1 - P_\mathrm{m})^{n-\delta(\mathbf{x},\mathbf{y})}$ строит строку $\mathbf{y}$ при мутации $\mathbf{x}$. Заметим, что вероятность $P_\mathrm{m}^k(1-P_\mathrm{m})^{n-k}$, как функция от P_m, $P_\mathrm{m} \in [0,1]$, достигает своего максимума при $P_\mathrm{m} = k/n$. Следующее утверждение дает оценку снизу

вероятности $\mathbf{P}\{\mathrm{Mut}^*(\mathbf{x}) = \mathbf{y}\}$ для любого $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ при оптимальном в указанном смысле выборе $P_\mathrm{m} = k/n$.

Утверждение 2.1. *Пусть система окрестностей $\mathcal{N}$ является k-ограниченной и $k \le n/2$. Тогда при $P_\mathrm{m} = k/n$ для любого $\mathbf{x} \in \mathrm{Sol}$ и любого $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ выполнено*

$$\mathbf{P}\{\mathrm{Mut}^*(\mathbf{x}) = \mathbf{y}\} \ge \left(\frac{k}{en}\right)^k.$$

Доказательство. При $\mathbf{x} \in \mathrm{Sol}$ и $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ имеем

$$\mathbf{P}\{\mathrm{Mut}^*(\mathbf{x}) = \mathbf{y}\} = \left(\frac{k}{n}\right)^{\delta(\mathbf{x},\mathbf{y})} \left(1 - \frac{k}{n}\right)^{n-\delta(\mathbf{x},\mathbf{y})} \ge \left(\frac{k}{n}\right)^k \left(1 - \frac{k}{n}\right)^{n-k},$$

так как $P_\mathrm{m} = k/n \le 1/2$. Далее, $\partial(1 - k/n)^{n-k}/\partial n < 0$ при $n > k$, кроме того, $(1-k/n)^{n-k} \to 1/e^k$ при $n \to \infty$. Следовательно, $(1-k/n)^{n-k} \ge 1/e^k$, откуда вытекает требуемое неравенство. □

2.1.2. Среднее время получения оптимума в двух специальных семействах задач

В качестве примера применения леммы 2.1 и утверждения 2.1 рассмотрим два семейства задач, где все локальные оптимумы являются глобальными.

Семейство одноэкстремальных задач ONEMAX**. Рассмотрим класс целевых функций ONEMAX**, введенный в [169]. Для этого определим сначала более узкий класс ONEMAX*, в который входят функции $\varphi_\eta : \{0,1\}^n \to \mathbb{Z}_+$ вида

$$\varphi_\eta(\xi) = n - \delta(\xi, \eta),$$

где $\eta \in \{0,1\}^n$. Тогда ONEMAX** определяется как класс функций $\psi \circ \varphi$, где $\varphi \in$ ONEMAX*, а $\psi : \mathbb{Z}_+ \to \mathbb{R}$ – строго возрастающая функция.

Индивидуальная задача принадлежит семейству ONEMAX**, если $f(\mathbf{x}) \in$ ONEMAX** и Sol $= \{0,1\}^n$. В качестве функции приспособленности для такой задачи естественно выбрать $\Phi(\mathbf{x}) \equiv f(\mathbf{x})$.

В системе окрестностей, порожденной метрикой Хэмминга радиуса 1, точка a является единственным локальным оптимумом, а значит, и глобальным. Согласно утверждению 2.1, при $P_{\mathrm{m}} = 1/n$, $n > 1$, для любого $\mathbf{x} \in$ Sol и любого $\mathbf{y} \in \mathcal{N}(\mathbf{x})$ выполнено $\mathbf{P}\{\mathrm{Mut}^*(\mathbf{x}) = \mathbf{y}\} \geq 1/(en)$. По лемме 2.1 заключаем, что при $s = rN$, $r > 0$, и

$$N = \left\lceil \frac{2en(1+\ln n)}{\varepsilon(1-1/e^{2r})} \right\rceil,$$

GA впервые посещает оптимум в среднем не позднее, чем за en итераций. Трудоемкость турнирной селекции такого ГА составляет $O(N)$. Если предположить, что кроссинговер, мутация и вычисление целевой функции могут быть выполнены за некоторое время T, то GA впервые посещает оптимум за время $O(Tn^3 \ln^2 n)$ в среднем. В классическом ГА [213], например, $T = O(n)$, т. е. оптимум впервые достигается в среднем за время $O(n^4 \ln^2 n)$.

Семейство задач вершинного покрытия графа $G(k)$. Рассмотрим семейство ЗВП специального вида, где граф $G(k)$ состоит из k трехвершинных клик, не связанных между собой, $k = 1, 2, \ldots$. Несмотря на простоту данной задачи, как показано в [91], некоторые известные алгоритмы типа ветвей и границ имеют экспоненциально возрастающую трудоемкость с увеличением числа вершин графа.

Пусть вершины и ребра графа пронумерованы, а каждому ребру $e_i \in E$ при кодировке решений из множества Sol сопоставлен один бит x_i, $i = 1, \ldots, n$, $n = |E|$. При таком способе представления решений любая строка $\mathbf{x} \in B$ кодирует допустимое решение, т. е. $C(\mathbf{x})$ является покрытием.

Пусть $\Phi(\mathbf{x}) = |V| - |C(\mathbf{x})|$. Таким образом, для $G(k)$ значение $\Phi(\mathbf{x})$ совпадает с количеством клик, оптимально покрытых вершинами $C(\mathbf{x})$.

В системе окрестностей, порожденной метрикой Хэмминга радиуса 1, любой локальный оптимум является и глобальным. Аналогично предыдущему примеру получаем $\mathbf{P}\{\mathrm{Mut}^*(\mathbf{x}) = \mathbf{y}\} \geq \frac{1}{en}$, где $n = 3k$. По лемме 2.1, при $s = rN$, где $r > 0$, и

$$N = \left\lceil \frac{6ek(1 + \ln k)}{\varepsilon(1 - 1/e^{2r})} \right\rceil,$$

GA впервые посещает оптимум в среднем не позднее, чем за ek итераций, т. е. среднее время его работы до получения оптимума полиномиально ограничено.

2.1.3. Задачи с гарантированными локальными оптимумами

Применим теорему 2.1 для оценки возможностей генетического алгоритма отыскивать приближенные решения с априорной оценкой точности.

Определение 2.2. [116] *Пусть* $\Pi \in \mathrm{NPO}$ – *полиномиально ограниченная задача.* Π *принадлежит классу задач с* гарантированными локальными оптимумами *(GLO), если следующие два условия выполняются:*

1) по крайней мере одно допустимое решение $\mathbf{y}_I \in \mathrm{Sol}$ *может быть вычислено за полиномиально ограниченное время для любой* $I \in \mathrm{Inst}$*;*

2) существует $k \in \mathbb{N}$ *такое, что все локальные оптимумы задачи* Π *относительно некоторой* k*-ограниченной системы окрестностей имеют константную оценку точности.*

Примерами задач из класса GLO являются задача о максимальной выполнимости логической формулы, задачи о наибольшем независимом множестве, о наименьшем доминирующем множестве вершин и о наименьшем вершинном покрытии в графах со степенью вершин, ограниченной константой, а также задача о максимальном разрезе графа [116].

Если задача комбинаторной оптимизации Π лежит в классе GLO, то ввиду утверждения 2.1, при $k \leq n/2$, для любого $\mathbf{x} \in \mathrm{Sol}$ и любого

$\mathbf{y} \in \mathcal{N}(\mathbf{x})$, оператор мутации Mut^* при $P_\mathrm{m} = k/n$ удовлетворяет условию $\mathbf{P}\{\mathrm{Mut}^*(\mathbf{x}) = \mathbf{y}\} \geq 1/\mathrm{poly}(|x|)$, где poly – некоторый полином. При $n/2 < k < n$ вероятность $\mathbf{P}\{\mathrm{Mut}^*(\mathbf{x}) = \mathbf{y}\}$ ограничена снизу положительной константой. Таким образом, из теоремы 2.1 вытекает следующее

Следствие 2.1. *Пусть* $\Pi \in \mathrm{GLO}$, $n > k$ *и популяция* X^0 *содержит допустимое решение. Тогда при операторе мутации* Mut^* *и соответствующем выборе параметров GA решение с константной оценкой точности впервые достигается в среднем за полиномиально ограниченное время.*

2.1.4. Заключение

Полученные оценки трудоемкости и предложенные значения настраиваемых параметров позволяют судить о тенденциях генетических алгоритмов, связанных с ростом размера турнира и численности популяции. На целесообразность увеличения размера турнира для скорейшего достижения локального оптимума указывают и результаты следующего раздела.

В настоящем разделе не учитывался тот факт, что в результате действия кроссинговера приспособленность потомков может оказаться выше приспособленности родителей. Улучшение известных теоретических оценок для ГА за счет учета такой возможности является открытой задачей. Обнадеживающие результаты в этом направлении получены в [157, 220].

2.2. Динамика численности особей с высокой приспособленностью в популяции генетического алгоритма

Настоящий раздел посвящен оценкам численности особей достаточно высокого качества на заданной итерации генетического алгоритма с турнирной селекцией. Рассматривается ГА с полной заменой популяции и турнирной селекцией без оператора кроссинговера. Для анализа ГА предлагается математическая модель эволюционного процесса, возникающего при

его работе. С использованием данной модели строятся нижние и верхние оценки среднего числа особей с приспособленностью не ниже заданного порогового значения. Установлен класс одноэкстремальных функций, на котором полученные оценки оказываются достижимы.

В литературе предложено несколько подходов к анализу ГА с использованием цепей Маркова. Например, при отождествлении состояний цепи со всевозможными векторами генотипов популяции X возникает цепь Маркова с 2^{lN} состояниями, если $\mathcal{B} = \{0,1\}^l$. С использованием этой модели для КГА показано Г. Рудольфом [276], что при бесконечном времени работы алгоритма генотипы, соответствующие оптимальному решению, будут появляться и исчезать в популяции бесконечно много раз. Аналогичный результат может быть получен для ГА с турнирной селекцией.

В другой модели [251], представляющей работу ГА с помощью цепи Маркова, все популяции, отличающиеся лишь способом упорядочения особей, считаются эквивалентными. Каждое состояние можно представить в виде вектора, имеющего 2^l компонент, где каждая координата отвечает одному из возможных генотипов. Значение компоненты равно доле, которую составляют в популяции особи с соответствующим этой компоненте генотипом. В рамках данной модели найдена переходная матрица цепи Маркова, отвечающей ГА с полной заменой популяции и пропорциональной селекцией для некоторых вариантов кроссинговера [290], и получен ряд важных результатов, описывающих поведение ГА [251, 289].

Основные трудности практического применения указанных моделей для анализа задач комбинаторной оптимизации вызваны необходимостью использования априорной информации о приспособленности каждого генотипа из $\mathcal{B}$ и большой размерностью матрицы перехода цепи Маркова. В п. 2.2.1 рассматривается способ преодоления этих трудностей посредством группировки состояний. Полученная модель не является цепью Маркова, однако позволяет найти нижние и верхние оценки числа особей с приспо-

собленностью не ниже заданного уровня на итерации t.

2.2.1. Оценки доли особей с заданной приспособленностью

Модель эволюционного процесса

В настоящем разделе предлагается модель эволюционного процесса, возникающего при работе ГА с полной заменой популяции при турнирной селекции и отсутствии кроссинговера, т. е. когда оператор кроссинговера представляет собой тождественное отображение. Для применения данной модели не требуется непосредственного использования информации о приспособленности каждого генотипа, как в [251, 276], однако предполагается, что известны некоторые оценки параметров оператора мутации.

При отсутствии кроссинговера в общей схеме ГА с полной заменой популяции нет необходимости генерировать особи каждой новой популяции обязательно парами, как в алгоритме 1.1. В связи с этим для большей общности здесь будем предполагать, что условие о четности N отсутствует и ГА имеет следующую схему.

Алгоритм 2.1. ГА без кроссинговера с полной заменой популяции
1. Положить $t := 0$.

2. Для k от 1 до N выполнять:

2.1. Построить случайным образом особь $\xi^{k,0}$.

Итерация t.

3. Для k от 1 до N выполнять шаги 3.1–3.2:

3.1. Селекция: выбрать особь $\xi := \xi^{\mathrm{Sel}(X^t),t}$.

3.2. Мутация: положить $\xi^{k,t+1} := \mathrm{Mut}(\xi)$.

4. Положить $t := t + 1$.

5. Если условие остановки выполнено, то идти на шаг 6, иначе – на шаг 3.

6. Результатом работы является особь с наибольшей приспособленностью среди найденных за все итерации.

Пусть $\Phi_0 = \min_{\xi \in \mathcal{B}} \Phi(\xi)$ и заданы линии уровня функции приспособленности $\Phi_1, \ldots, \Phi_d$, такие что

$$\Phi_0 < \Phi_1 < \Phi_2 \ldots < \Phi_d.$$

Число линий уровня и соответствующие им значения функции приспособленности (кроме Φ_0) могут быть выбраны достаточно произвольно с учетом свойств решаемой задачи и оператора мутации. Далее будем предполагать, что число линий уровня, а также их значения некоторым образом зависят от индивидуальной задачи I. Для простоты обозначений линий уровня символ I, как правило, не будет указываться при их записи.

Сопоставим выбранным линиям уровня следующие подмножества:

$$H_i = \{\xi : \Phi(\xi) \geq \Phi_i\}, \quad i = 0, \ldots, d.$$

Заметим, что $H_0 = \mathcal{B}$. Для удобства обозначений положим $H_{d+1} = \emptyset$.

Пусть для всех $i = 0, ..., d$ и $j = 1, ..., d$ априори известны нижние α_{ij} и верхние β_{ij} оценки вероятности перехода из $H_i \backslash H_{i+1}$ в H_j при действии оператора мутации, а именно для любого $\xi \in H_i \backslash H_{i+1}$

$$\alpha_{ij} \leq \mathbf{P}\{\mathrm{Mut}(\xi) \in H_j\} \leq \beta_{ij},$$

где

$$\mathbf{P}\{\mathrm{Mut}(\xi) \in H_j\} = \sum_{\xi' \in H_j} \mathbf{P}\{\mathrm{Mut}(\xi) = \xi'\}.$$

Обозначим матрицу с элементами α_{ij}, где $i = 0, ..., d, j = 1, ..., d$ через $\mathbf{A}$; аналогичную матрицу верхних оценок обозначим через $\mathbf{B}$.

Представим популяцию на шаге t при помощи *вектора популяции*

$$\mathbf{z}^{(t)} = (z_1^{(t)}, z_2^{(t)}, \ldots, z_d^{(t)}),$$

где $z_i^{(t)} \in \mathbb{R}$ - доля генотипов из H_i в популяции X^t. Таким образом, $\mathbf{z}^{(t)}$ – случайный вектор, причем $z_i^{(t)} \geq z_{i+1}^{(t)}$ для $i = 1, ..., d-1$, так как $H_{i+1} \subseteq H_i$. Далее для удобства будем полагать $z_{d+1} = z_{d+1}^{(t)} = 0$ при любом t.

Пусть $\mathbf{P}\{\xi^{(t)} \in H_j\}$ – вероятность того, что генотип, полученный в результате селекции из популяции с номером $t-1$, после мутации будет принадлежать H_j. Заметим, что все особи поколения t в рассматриваемом алгоритме генерируются с одним и тем же распределением вероятностей, следовательно, $\mathbf{P}\{\xi^{(t)} \in H_j\} = \mathbf{P}\{\xi_1^{(t)} \in H_j\} = ... = \mathbf{P}\{\xi_N^{(t)} \in H_j\}$ при любом $t > 0$ и $j = 1, ..., d$. Справедливо следующее

Утверждение 2.2. *При любом* $t > 0$ *имеет место равенство* $\mathbf{E}[z_i^{(t)}] = \mathbf{P}\{\xi^{(t)} \in H_i\}$.

Доказательство. Рассмотрим последовательность одинаково распределенных случайных величин $\mathcal{I}_1^i, \mathcal{I}_2^i, ..., \mathcal{I}_N^i$, где $\mathcal{I}_k^i = 1$, если k-я особь популяции X^t лежит в H_i, и $\mathcal{I}_k^i = 0$ в противном случае. По определению вектора популяции имеем $z_i^{(t)} = \sum_{k=1}^N \mathcal{I}_k^i / N$, следовательно,

$$\mathbf{E}[z_j^{(t)}] = \sum_{k=1}^{N} \frac{\mathbf{E}[\mathcal{I}_k^i]}{N} = \sum_{k=1}^{N} \frac{\mathbf{P}\{\xi^{(t)} \in H_i\}}{N} = \mathbf{P}\{\xi^{(t)} \in H_i\}.$$

Получим нижние и верхние оценки доли особей с генотипами из подмножеств H_i для $i = 1, \ldots, d$ на заданной итерации.

Обозначим через $\mathbf{P}_{\text{ch}}(S, \mathbf{z})$ вероятность того, что генотип, выбранный при помощи турнирной селекции из популяции с вектором $\mathbf{z}$, принадлежит подмножеству $S \subseteq B$. Заметим, что если текущая популяция представляется вектором $\mathbf{z}^{(t)} = \mathbf{z}$, то условная вероятность того, что очередной генотип, полученный в результате селекции и мутации, принадлежит H_j, имеет вид:

$$\mathbf{P}\{\xi^{(t+1)} \in H_j | \mathbf{z}^{(t)} = \mathbf{z}\} = \sum_{i=0}^{d} \sum_{\xi \in H_i \setminus H_{i+1}} \mathbf{P}\{\text{Mut}(\xi) \in H_j\} \mathbf{P}_{\text{ch}}(\{\xi\}, \mathbf{z}). \quad (2.9)$$

Нижние оценки

Из формулы (2.9) и определения оценок α_{ij} следует, что

$$\mathbf{P}\{\xi^{(t+1)} \in H_j | \mathbf{z}^{(t)} = \mathbf{z}\} \geq \sum_{i=0}^{d} \alpha_{ij} \sum_{\xi \in H_i \setminus H_{i+1}} \mathbf{P}_{\text{ch}}(\{\xi\}, \mathbf{z}) =$$

$$= \sum_{i=0}^{d} \alpha_{ij} \mathbf{P}_{\text{ch}}(H_i \backslash H_{i+1}, \mathbf{z}).$$

При размере турнира s имеем $\mathbf{P}_{\text{ch}}(H_i, \mathbf{z}^{(t)}) = 1 - (1 - z_i^{(t)})^s$, и, следовательно, $\mathbf{P}_{\text{ch}}(H_i \backslash H_{i+1}, \mathbf{z}) = \mathbf{P}_{\text{ch}}(H_i, \mathbf{z}) - \mathbf{P}_{\text{ch}}(H_{i+1}, \mathbf{z}) = (1 - z_{i+1})^s - (1 - z_i)^s$. Отсюда получаем оценку снизу:

$$\mathbf{P}\{\xi^{(t+1)} \in H_j | \mathbf{z}^{(t)} = \mathbf{z}\} \geq \sum_{i=0}^{d} \alpha_{ij}((1 - z_{i+1})^s - (1 - z_i)^s).$$

Пусть $Z_N = \{\mathbf{z} \in \mathbb{R}^d : z_i \in \{0, \frac{1}{N}, \frac{2}{N}, \ldots, 1\}, z_i \geq z_{i+1}\}$ – множество векторов популяции из N особей. По формуле полной вероятности получаем безусловную вероятность порождения очередной особи из H_j на шаге $t+1$:

$$\mathbf{P}\{\xi^{(t+1)} \in H_j\} = \sum_{\mathbf{z} \in Z_N} \mathbf{P}\{\xi^{(t+1)} \in H_j | \mathbf{z}^{(t)} = \mathbf{z}\} \mathbf{P}\{\mathbf{z}^{(t)} = \mathbf{z}\} \geq \tag{2.10}$$

$$\geq \sum_{\mathbf{z} \in Z_N} \sum_{i=0}^{d} \alpha_{ij}((1 - z_{i+1})^s - (1 - z_i)^s) \mathbf{P}\{\mathbf{z}^{(t)} = \mathbf{z}\} =$$

$$= \sum_{i=0}^{d} \alpha_{ij} \mathbf{E}[(1 - z_{i+1}^{(t)})^s - (1 - z_i^{(t)})^s].$$

Ввиду утверждения 2.2 $\mathbf{E}[z_j^{(t+1)}] = \mathbf{P}\{\xi^{(t+1)} \in H_j\}$. Следовательно, с учетом того, что $(1 - z_{d+1}^{(t)})^s = 1$ и $(1 - z_0^{(t)})^s = 0$, из (2.10) получаем:

$$\mathbf{E}[z_j^{(t+1)}] \geq \alpha_{dj} - \sum_{i=1}^{d} (\alpha_{i,j} - \alpha_{i-1,j}) \mathbf{E}[(1 - z_i^{(t)})^s]. \tag{2.11}$$

Пусть $I_j^+(\mathbf{A}) = \{i : 1 \leq i \leq d,\ \alpha_{i,j} - \alpha_{i-1,j} \geq 0\}$, $I_j^-(\mathbf{A}) = \{i : 1 \leq i \leq d,\ \alpha_{i,j} - \alpha_{i-1,j} < 0\}$. Применим к тем слагаемым последней суммы, где $i \in I_j^-(\mathbf{A})$, неравенство Иенсена (под знаком математического ожидания стоит выпуклая функция от $z_i^{(t)}$). Для слагаемых, где $i \in I_j^+(\mathbf{A})$, можно воспользоваться оценкой $(1 - z_i^{(t)})^s \leq 1 - z_i^{(t)}$. В результате имеем

Утверждение 2.3. *Математическое ожидание компонента j вектора*

популяции $\mathbf{z}^{(t+1)}$ *удовлетворяет неравенству*

$$\mathbf{E}[z_j^{(t+1)}] \geq \alpha_{dj} - \sum_{I_j^+(\mathbf{A})} (\alpha_{ij} - \alpha_{i-1,j})(1 - \mathbf{E}[z_i^{(t)}]) - \sum_{I_j^-(\mathbf{A})} (\alpha_{ij} - \alpha_{i-1,j})(1 - \mathbf{E}[z_i^{(t)}])^s \tag{2.12}$$

для всех $j = 1, \ldots, d$.

Заметим, что если при всех $i = 0, ..., d$, $j = 1, ..., d$, вероятность $\mathbf{P}\{\mathrm{Mut}(\xi) \in H_j\}$ не зависит от выбора $\xi \in H_i \backslash H_{i+1}$, то всегда можно указать такие матрицы оценок $\mathbf{A}$ и $\mathbf{B}$, что $\alpha_{ij} = \beta_{ij} = \mathbf{P}\{\mathrm{Mut}(\xi) \in H_j | \xi \in H_i \backslash H_{i+1}\}$ для всех i и j. Условимся в таком случае называть оператор мутации *ступенчатым* относительно набора линий уровня $\Phi_1, ..., \Phi_d$ (или просто ступенчатым, если ясно, о каком наборе подмножеств идет речь). Матрицу $\mathbf{M} = \mathbf{A} = \mathbf{B}$ с элементами μ_{ij}, $i = 0, \ldots, d$, $j = 1, \ldots, d$, дающую одновременно верхнюю и нижнюю оценки для ступенчатого оператора, будем называть матрицей *кумулятивных переходных вероятностей* (по аналогии с матрицей переходных вероятностей цепи Маркова).

При ступенчатом операторе мутации эволюционный процесс в ГА моделируется цепью Маркова $\{\mathbf{z}^{(t)} : t = 0, 1, \ldots\}$, состояниями которой являются всевозможные векторы популяции $\mathbf{z}^{(t)} \in Z_N$. Для того чтобы в этом убедиться, рассмотрим цепь Маркова $\{X^t : t = 0, 1, \ldots\}$, где состояниями могут быть всевозможные векторы генотипов популяции $X^t \in \mathcal{B}$. Пусть $X(1)$ и $X(2)$ - две популяции, имеющие одинаковое представление вектором $\mathbf{z}^{(t)}$. Обозначим через $\mathbf{P}_{\mathrm{ch}}(S, X(k))$ вероятность того, что генотип, выбранный при селекции из популяции $X(k)$, принадлежит подмножеству $S \subseteq \mathcal{B}$. Тогда по аналогии с выводом формулы (2.9), используя ступенчатость мутации, для любых i и j от 0 до d получаем:

$$\mathbf{P}\{\xi^{(t+1)} \in H_j | X^t = X(k)\} = \sum_{i=0}^{d} \mu_{ij} \sum_{\xi \in H_i \backslash H_{i+1}} \mathbf{P}_{\mathrm{ch}}(\{\xi\}, X(k)) =$$

$$= \sum_{i=0}^{d} \mu_{ij} \mathbf{P}_{\mathrm{ch}}(H_i \backslash H_{i+1}, X(k)), \quad k = 1, 2.$$

Согласно схеме ГА с турнирной селекцией вероятности $\mathbf{P}_{\mathrm{ch}}(H_i \backslash H_{i+1}, X(k))$ полностью определяются вектором $\mathbf{z}^{(t)}$ и не зависят от k. Следовательно, при ступенчатом операторе мутации распределение вектора $\mathbf{z}^{(t+1)}$ также не имеет зависимости от k. Отсюда, пользуясь укрупнением состояний в цепи Маркова (см., например, [58]), приходим к выводу о том, что рассматриваемый ГА корректно моделируется цепью Маркова $\{\mathbf{z}^{(t)} : t = 0, 1, \ldots\}$.

Для получения оценки величины $\mathbf{E}[\mathbf{z}^{(t)}]$ при известном векторе средних начальной популяции $\mathbf{E}[\mathbf{z}^{(0)}]$ потребуется t-кратное применение неравенства (2.12). С этой целью введем следующее определение.

Матрицу размерности $d \times d$ с элементами μ_{ij} будем называть *монотонной*, если $\mu_{i-1,j} \leq \mu_{i,j}$ для всех i, j от 1 до d. Ступенчатый оператор мутации относительно набора линий уровня $\Phi_1, ..., \Phi_d$ будем называть *монотонным* относительно этого набора линий уровня, если его матрица кумулятивных вероятностей перехода монотонна. Оператор называется *монотонным*, если данное свойство выполнено хотя бы при одном наборе линий уровня. (В соответствии с терминологией Д. Дэлей [152] такой оператор является *стохастически монотонным*.)

Очевидно, что для любого оператора мутации существуют монотонные оценки (например, нулевая матрица $\mathbf{A}$ и матрица $\mathbf{B}$ со всеми элементами, равными 1). Однако трудности могут возникать только из-за отсутствия оценок, достаточно точно передающих свойства этого оператора.

После несложных преобразований формулы (2.12) получаем

Утверждение 2.4. *При условии монотонности матрицы* $\mathbf{A}$ *для всех* $j = 1, \ldots, d$ *имеет место оценка*

$$\mathbf{E}[z_j^{(t+1)}] \geq \alpha_{0j} + \sum_{i=1}^{d} (\alpha_{ij} - \alpha_{i-1,j}) \mathbf{E}[z_i^{(t)}]. \tag{2.13}$$

Ввиду того, что оценка (2.13) не зависит от N и s, величина $\mathbf{E}[z_k^{(t)}]$, $k = 1, \ldots, d,\ t = 0, 1...$ при произвольных N и s оценивается снизу значением вероятности $\mathbf{P}\{\xi^{(t)} \in H_k\}$, найденным для ГА при $N = 1$, $s = 1$, где оператор мутации имеет кумулятивные переходные вероятности, совпадающие с монотонными оценками α_{ij}, $i = 0, \ldots, d,\ j = 1, \ldots, d$. Значения таких вероятностей $\mathbf{P}\{\xi^{(t)} \in H_k\}$ могут быть найдены методами анализа обобщенных одномерных случайных блужданий (см., например, [100], гл. 4, § 8), применимыми к цепи Маркова $\{\mathbf{z}^{(t)} : t = 0, 1, \ldots\}$, так как вектор популции при $N = 1$ имеет вид $(1, \ldots, 1, 0, \ldots, 0)$.

Для получения оценки снизу в достаточно общей ситуации может быть использовано следующее утверждение. Пусть $\mathbf{W}$ – $(d \times d)$-матрица с элементами $w_{ij} = \alpha_{ij} - \alpha_{i-1,j}$, $i = 1, \ldots, d,\ j = 1, \ldots, d$; $\mathbf{I}$ – единичная матрица того же размера; вектор $\mathbf{a} = (\alpha_{01}, ..., \alpha_{0d})$; $||\cdot||$ – произвольная матричная норма.

Утверждение 2.5. *Если матрица* $\mathbf{A}$ *монотонна и* $||\mathbf{W}^t|| \stackrel{t\to\infty}{\longrightarrow} 0$, *то для любого натурального* t *имеет место оценка*

$$\mathbf{E}[\mathbf{z}^{(t)}] \geq \mathbf{E}[\mathbf{z}^{(0)}]\mathbf{W}^t + \mathbf{a}(\mathbf{I} - \mathbf{W})^{-1}(\mathbf{I} - \mathbf{W}^t). \tag{2.14}$$

Доказательство. Рассмотрим в $\mathbb{R}^d$ последовательность векторов $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, ..., \mathbf{u}^{(t)}, ...$, где $\mathbf{u}^{(0)} = \mathbf{E}[\mathbf{z}^{(0)}]$, $\mathbf{u}^{(t+1)} = \mathbf{u}^{(t)}\mathbf{W} + \mathbf{a}$. Правая часть неравенства (2.13) может только уменьшаться при подстановке туда нижних оценок компонентов вектора $\mathbf{E}[\mathbf{z}^{(t)}]$, поэтому по индукции для любого t получаем: $\mathbf{E}[\mathbf{z}^{(t)}] \geq \mathbf{u}^{(t)}$.

Пользуясь свойствами линейных операторов, ввиду условия $||\mathbf{W}^t|| \stackrel{t\to\infty}{\longrightarrow} 0$ заключаем, что матрица $(\mathbf{I} - \mathbf{W})^{-1}$ существует.

Далее по индукции с помощью непосредственной проверки легко показать, что для любого $t \geq 1$ имеет место тождество

$$\mathbf{u}^{(t)} = \mathbf{u}^{(0)}\mathbf{W}^t + \mathbf{a}(\mathbf{I} - \mathbf{W})^{-1}(\mathbf{I} - \mathbf{W}^t).$$

□

Правая часть в (2.14) с ростом t приближается к $\mathbf{a}(\mathbf{I}-\mathbf{W})^{-1}$, поэтому предел оценки (2.14) не зависит от распределения популяции X^0. При ступенчатом операторе мутации и $s=1$ оценка (2.14) обращается в равенство.

Пусть $\alpha_{dj}-\alpha_{0j}<1$, $j=1,\ldots,d$. Рассмотрим матричную норму $||\mathbf{W}||=\max_j\sum_{i=1}^d|w_{ij}|$. В условиях теоремы $w_{ij}=\alpha_{ij}-\alpha_{i-1,j}\geq 0$, поэтому $||\mathbf{W}||=\max_j\sum_{i=1}^d w_{ij}=\max_j(\alpha_{dj}-\alpha_{0j})<1$, и условие $||\mathbf{W}^t||\overset{t\to\infty}{\longrightarrow}0$ выполнено. Заметим, что $\alpha_{dj}-\alpha_{0j}<1$ для всех j, если при мутации из любого генотипа с ненулевой вероятностью может быть получен любой наперед заданный генотип.

С использованием того же подхода, что и в утверждении 2.5, в [135, 136] нами получены нижние и верхние оценки вероятности получения генотипа из множества H_j, $j=1,\ldots,d$, на заданной итерации эволюционных стратегий (1+1)-ES и (1,λ)-ES. Также в этих работах исследуется, насколько велики могут быть нижние оценки α_{ij} для полиномиально вычислимого оператора мутации при решении полиномиально ограниченной NP-трудной задачи, если верна известная гипотеза NP $\not\subseteq$ BPP (см., например, [59]).

Оценки сверху

По аналогии с выводом формулы (2.10) получаем оценку сверху:

$$\mathbf{E}[z_j^{(t+1)}]\leq\beta_{dj}-\sum_{i=1}^d(\beta_{ij}-\beta_{i-1,j})\mathbf{E}[(1-z_i^{(t)})^s]. \tag{2.15}$$

Отсюда вытекает следующее

Утверждение 2.6. *Математическое ожидание вектора популяции удовлетворяет неравенству*

$$\mathbf{E}[z_j^{(t+1)}]\leq\beta_{dj}-\sum_{I_j^-(\mathbf{B})}(\beta_{ij}-\beta_{i-1,j})(1-\mathbf{E}[z_i^{(t)}])-\sum_{I_j^+(\mathbf{B})}(\beta_{ij}-\beta_{i-1,j})(1-\mathbf{E}[z_i^{(t)}])^s \tag{2.16}$$

для $j = 1, \ldots, d$, *и если матрица* $\mathbf{B}$ *монотонна, то*

$$\mathbf{E}[z_j^{(t+1)}] \leq \beta_{dj} - \sum_{i=1}^{d} (\beta_{ij} - \beta_{i-1,j})(1 - \mathbf{E}[z_i^{(t)}])^s, \quad j = 1, \ldots, d. \tag{2.17}$$

Посредством итеративного применения неравенства (2.17) можно покомпонентно оценить сверху вектор $\mathbf{E}[\mathbf{z}^{(t)}]$ для любого t, начав с вектора средних исходной популяции $\mathbf{E}[\mathbf{z}^{(0)}]$. Тем не менее, нелинейность правой части (2.17) затрудняет получение компактной оценки сверху для произвольного $t \geq 0$, аналогичной неравенству (2.14) из утверждения 2.5.

Полученные выше оценки не учитывают размер популяции и справедливы при любом N. Как будет показано в утверждении 2.7, правая часть формулы (2.17) описывает асимптотику поведения популяции при монотонном операторе мутации и $N \to \infty$.

Случай монотонного оператора мутации

При монотонном операторе мутации оценки (2.11) и (2.15) равны и

$$\mathbf{E}[z_j^{(t+1)}] = \mu_{dj} - \sum_{i=1}^{d} (\mu_{ij} - \mu_{i-1,j}) \mathbf{E}[(1 - z_i^{(t)})^s], \quad j = 1, \ldots, d, \quad t = 0, 1, \ldots. \tag{2.18}$$

Данное равенство будет использоваться далее неоднократно.

Вообще говоря, векторы популяций являются случайными величинами, распределение которых зависит от N. Для того чтобы отразить размер популяции в обозначениях, условимся записывать долю генотипов из H_i в популяции X^t через $z_i^{(t)}(N)$.

Лемма 2.2. *Пусть ГА имеет монотонный оператор мутации, а особи начальной популяции одинаково распределены. Тогда*

(i) для любых $t = 0, 1, \ldots$ *и* $i = 1, \ldots, d$ *имеет место соотношение*

$$\lim_{N \to \infty} \left(\mathbf{E}\left[\left(1 - z_i^{(t)}(N) \right)^s \right] - \left(1 - \mathbf{E}[z_i^{(t)}(N)] \right)^s \right) = 0, \tag{2.19}$$

(ii) если последовательность d-мерных векторов $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, ..., \mathbf{u}^{(t)}, ...$ определена равенствами

$$\mathbf{u}^{(0)} = \mathbf{E}[\mathbf{z}^{(0)}(N)], \tag{2.20}$$

$$u_j^{(t+1)} = \mu_{dj} - \sum_{i=1}^{d} (\mu_{ij} - \mu_{i-1,j})(1 - u_i^{(t)})^s \tag{2.21}$$

для всех $j = 1, ..., d$, то для любого t имеем: $\mathbf{E}[\mathbf{z}^{(t)}(N)] \underset{N\to\infty}{\longrightarrow} \mathbf{u}^{(t)}$.

Доказательство. Ввиду (2.18) заключаем, что если будет доказан пункт (i) настоящей леммы, то при $N \to \infty$ из сходимости $\mathbf{E}[\mathbf{z}^{(t)}(N)]$ к $\mathbf{u}^{(t)}$ будет следовать сходимость $\mathbf{E}[\mathbf{z}^{(t+1)}(N)]$ к $\mathbf{u}^{(t+1)}$. В таком случае пункт (ii) доказывается индукцией по t для любого конечного t.

При фиксированном t для доказательства формулы (2.19) вернемся к рассматривавшейся ранее последовательности одинаково распределенных случайных величин $\mathcal{I}_1^i, \mathcal{I}_2^i, ..., \mathcal{I}_N^i$, где $\mathcal{I}_k^i$ равна единице, если k-я особь популяции X^t лежит в H_i, и нулю в противном случае. Применяя закон больших чисел для любых $i = 1, ..., d$ и $\varepsilon > 0$, имеем

$$\mathbf{P}\left\{\left|\frac{\sum_{k=1}^N \mathcal{I}_k^i}{N} - \mathbf{E}[\mathcal{I}_1^i]\right| < \varepsilon\right\} \underset{N\to\infty}{\longrightarrow} 1.$$

Заметим, что $\sum_{k=1}^N \mathcal{I}_k^i/N = z_i^{(t)}(N)$. Кроме того, ввиду утверждения 2.2 $\mathbf{E}[\mathcal{I}_1^i] = \mathbf{P}\{\mathcal{I}_1^i = 1\} = \mathbf{E}[z_i^{(t)}(N)]$ (в случае $t = 0$, это равенство также справедливо ввиду того, что все особи начальной популяции распределены одинаково). Следовательно, для любого $\varepsilon > 0$ имеет место сходимость $\mathbf{P}\left\{\left|z_i^{(t)}(N) - \mathbf{E}[z_i^{(t)}(N)]\right| < \varepsilon\right\} \underset{N\to\infty}{\longrightarrow} 1$. Отсюда, ввиду непрерывности функции $(1-x)^s$, заключаем, что

$$\mathbf{P}\left\{\left|(1 - z_i^{(t)}(N))^s - (1 - \mathbf{E}[z_i^{(t)}(N)])^s\right| \geq \varepsilon\right\} \underset{N\to\infty}{\longrightarrow} 0.$$

Обозначим $\mathrm{F}_N(x) = \mathbf{P}\left\{(1 - z_i^{(t)}(N))^s - (1 - \mathbf{E}[z_i^{(t)}(N)])^s < x\right\}$. Тогда

$$\lim_{N\to\infty}\left(\mathbf{E}\left[(1 - z_i^{(t)}(N))^s\right] - (1 - \mathbf{E}[z_i^{(t)}(N)])^s\right) = \lim_{N\to\infty} \int_{-\infty}^{\infty} x\, d\mathrm{F}_N(x) \leq$$

$$\leq \lim_{N\to\infty} \mathbf{P}\left\{\left|(1-z_i^{(t)}(N))^s - (1-\mathbf{E}[z_i^{(t)}(N)])^s\right| \geq \varepsilon\right\} + \lim_{N\to\infty} \int\limits_{|x|<\varepsilon} \varepsilon\, d\mathrm{F}_N(x) = \varepsilon.$$

для произвольного $\varepsilon > 0$, следовательно, формула (2.19) верна. $\square$

Ввиду равенств (2.18) и (2.19) из леммы 2.2 вытекает следующее

Утверждение 2.7. *Пусть ГА имеет монотонный оператор мутации, а особи начальной популяции одинаково распределены. Тогда*

$$\mathbf{E}[z_j^{(t+1)}(N)] \underset{N\to\infty}{\longrightarrow} \mu_{dj} - \sum_{i=1}^{d}(\mu_{ij} - \mu_{i-1,j})(1-\mathbf{E}[z_i^{(t)}(N)])^s \qquad (2.22)$$

для всех $j = 1, \ldots, d,\ t \geq 0$.

Исследуем асимптотические свойства вектора популяции, связанные с изменением размера турнира. Сначала докажем вспомогательную лемму.

Лемма 2.3. *Если ГА имеет монотонный оператор мутации, последовательность векторов* $\{\mathbf{u}^{(t)}\}$ *определена согласно (2.20), (2.21) и* $u_j^{(t)} \in \{0,1\}$ *при некоторых* $j \in \{1,\ldots,d\}$ *и* $t \geq 0$, *тогда* $\mathbf{E}[z_j^{(t)}] = u_j^{(t)}$.

Доказательство. Докажем лемму для случая $u_j^{(t)} = 0$, пользуясь индукцией по t. При $t = 0$ утверждение верно по определению $\mathbf{u}^{(0)}$. Пусть для t утверждение верно, докажем его для $t+1$.

Для удобства обозначений положим $u_{d+1}^{(t)} = 0$ для всех t и определим номер k следующим образом:

$$k = \max\{i : 1 \leq i \leq d+1,\ u_i^{(t)} \neq u_{i-1}^{(t)}\}.$$

Тогда $\mu_{k-1,j} = 0$, так как из (2.21) имеем

$$0 = u_j^{(t+1)} = \sum_{i=1}^{d+1} \mu_{i-1,j}((1-u_i^{(t)})^s - (1-u_{i-1}^{(t)})^s),$$

где $u_i^{(t)} \leq u_{i-1}^{(t)}$. Последнее неравенство доказывается по индукции ввиду

$$u_j^{(t+1)} - u_{j-1}^{(t+1)} = \sum_{i=1}^{d+1}(\mu_{i-1,j} - \mu_{i-1,j-1})((1-u_i^{(t)})^s - (1-u_{i-1}^{(t)})^s).$$

Далее с использованием монотонности получаем

$$\mu_{0j} = \ldots = \mu_{k-1,j} = 0. \tag{2.23}$$

Из определения величины k следует, что $u_k^{(t)} = \ldots = u_{d+1}^{(t)} = 0$, а значит, по предположению индукции имеем также $\mathbf{E}[z_k^{(t)}] = \ldots = \mathbf{E}[z_{d+1}^{(t)}] = 0$. Отсюда, с учетом (2.18) и (2.23), получаем

$$\mathbf{E}[z_j^{(t+1)}(N)] = \sum_{i=1}^{d+1} \mu_{i-1,j}((1 - \mathbf{E}[z_i^{(t)}])^s - (1 - \mathbf{E}[z_{i-1}^{(t)}])^s) = 0.$$

Доказательство для случая $u_j^{(t)} = 1$ проводится аналогично. □

Теорема 2.2. *Пусть* $\mathbf{z}^{(t)}$ *и* $\hat{\mathbf{z}}^{(t)}$ *– векторы популяции ГА с турнирами величины* s *и* $\hat{s}$ *соответственно, причем* $s \leq \hat{s}$. *Тогда если ГА имеет монотонный оператор мутации и особи начальных популяций одинаково распределены, то для любых* $t = 0, 1, \ldots$ *и* $i = 1, ..., d$ *при достаточно большом размере популяции имеет место неравенство* $\mathbf{E}[z_i^{(t)}] \leq \mathbf{E}[\hat{z}_i{}^{(t)}]$.

Доказательство. При любых i, j и $t > 0$ элемент $u_j^{(t)}$ указанной в лемме 2.2 последовательности $\{u_j^{(t)}\}$ представляет собой неубывающую функцию как от $u_i^{(t-1)}$, так и от s. Более того, все элементы $0 < u_i^{(t)} < 1$ строго монотонно возрастают с ростом s. Для всех таких i и t при достаточно большом размере популяции выполняется неравенство $\mathbf{E}[\hat{z}_i{}^{(t)}(N)] > \mathbf{E}[z_i^{(t)}(N)]$, как следует из пункта (ii) леммы 2.2. В остальных случаях, т. е. когда $u_i^{(t)} \in \{0, 1\}$, утверждение теоремы вытекает из леммы 2.3. □

2.2.2. Примеры использования предложенной модели

Приведем несколько иллюстративных примеров ситуаций, когда достаточно просто получить кумулятивные вероятности перехода для оператора мутации или вычислить оценки этих вероятностей.

Пример 1. Предположим, что кодировка решений в ГА такова, что строка генотипа разбивается на d непересекающихся блоков и значение

функции приспособленности $\Phi(\xi)$ равно числу блоков, для которых выполнено некоторое свойство $\mathbf{K}$. Пусть m – число блоков и $K(\xi,\ell)=1$, если свойство $\mathbf{K}$ имеет место для блока с номером ℓ генотипа ξ; в противном случае $K(\xi,\ell)=0$. Рассмотрим оператор мутации $\mathrm{Mut}_{r\tilde{r}}$, такой что любой блок, для которого данное свойство не было выполнено, становится блоком со свойством $\mathbf{K}$ после мутации с вероятностью, равной $\tilde{r}$, т. е.

$$\mathbf{P}\{K(\mathrm{Mut}_{r\tilde{r}}(\xi),\ell)=1|K(\xi,\ell)=0\}=\tilde{r},\quad \ell=1,...,m,$$

с другой стороны, любой блок, для которого выполнено $\mathbf{K}$, сохраняет это свойство в процессе мутации с вероятностью r, т. е.

$$\mathbf{P}\{K(\mathrm{Mut}_{r\tilde{r}}(\xi),\ell)=1|K(\xi,\ell)=1\}=r,\quad \ell=1,...,m.$$

Утверждение 2.8. *Если $r\geq\tilde{r}$, то оператор мутации $\mathrm{Mut}_{r\tilde{r}}$ является монотонным.*

Доказательство. Пусть подмножества $H_0,...,H_d$ соответствуют линиям уровня $\Phi_0=0,\Phi_1=1,...,\Phi_d=d$. В таком случае μ_{ij} есть вероятность получить генотип, имеющий не менее j блоков со свойством $\mathbf{K}$, если до мутации этим свойством обладало i блоков.

Вероятность того, что из $d-i$ блоков, не обладающих $\mathbf{K}$, в результате мутации будет получено $k=0,...,d-i$ блоков со свойством $\mathbf{K}$ равна $\binom{d-i}{k}\tilde{r}^k(1-\tilde{r})^{d-i-k}$. С другой стороны, из i блоков, обладающих $\mathbf{K}$, данное свойство при мутации потеряют не более чем i' блоков с вероятностью

$$\sum_{\nu=0}^{\min\{i,i'\}}\binom{i}{\nu}(1-r)^{\nu}r^{i-\nu}$$

для любого $i'\geq 0$. Таким образом, данный ступенчатый оператор имеет кумулятивные вероятности перехода

$$\mu_{ij}=\sum_{k=0}^{d-i}\binom{d-i}{k}\tilde{r}^k(1-\tilde{r})^{d-i-k}\sum_{\nu=0}^{\min\{i,i-j+k\}}\binom{i}{\nu}(1-r)^{\nu}r^{i-\nu} \qquad (2.24)$$

при $i = 0, ..., d, j = 1, ..., d$.

Заметим, что если $r \geq \tilde{r}$, то матрица $\mathbf{M}$ с элементами μ_{ij}, определяемыми формулой (2.24), является монотонной. Действительно, рассмотрим действие данного оператора мутации на двух разных генотипах $\xi \in H_i$ и $\xi' \in H_{i-1}$. Пусть $\{\mathcal{I}_k\}$ и $\{\mathcal{I}'_k\}$ – две последовательности независимых одинаково распределенных величин, принимающих значение 1 с вероятностью r и равных 0 в противном случае. Пусть $\{\mathcal{J}_k\}$ и $\{\mathcal{J}'_k\}$ также являются последовательностями независимых одинаково распределенных величин, равных единице с вероятностью $\tilde{r}$ и нулю в противном случае. Тогда по определению μ_{ij},

$$\mu_{ij} - \mu_{i-1,j} =$$
$$\mathbf{P}\left\{\sum_{k=1}^{i-1}\mathcal{I}_k + \mathcal{I}_i + \sum_{k=1}^{d-i}\mathcal{J}_k \geq j\right\} - \mathbf{P}\left\{\sum_{k=1}^{i-1}\mathcal{I}'_k + \sum_{k=1}^{d-i}\mathcal{J}'_k + \mathcal{J}'_{d-i+1} \geq j\right\}.$$

Отсюда, применяя формулу полной вероятности, с учетом независимости $\mathcal{I}_i$ от $\mathcal{I}_1, ..., \mathcal{I}_{i-1}$ и $\mathcal{J}_1, ..., \mathcal{J}_{d-i}$, а также независимости $\mathcal{J}'_{d-i+1}$ от $\mathcal{I}'_1, ..., \mathcal{I}'_{i-1}$ и $\mathcal{J}'_1, ..., \mathcal{J}'_{d-i}$, разность $\mu_{ij} - \mu_{i-1,j}$ можно записать в виде

$$\sum_{\theta=0}^{d-1}\left[\mathbf{P}\left\{\sum_{k=1}^{i-1}\mathcal{I}_k + \sum_{k=1}^{d-k}\mathcal{J}_k = \theta\right\}\left(\mathbf{P}\{\mathcal{I}_i \geq j - \theta\} - \mathbf{P}\{\mathcal{J}'_{d-i+1} \geq j - \theta\}\right)\right],$$

что не может быть отрицательным, так как

$$\mathbf{P}\{\mathcal{I}_i \geq j - \theta\} - \mathbf{P}\{\mathcal{J}'_{d-i+1} \geq j - \theta\} \geq 0$$

для любых j и θ, если $r \geq \tilde{r}$. $\square$

В качестве примера применения утверждения 2.8 рассмотрим семейство задач вершинного покрытия графа специального вида $G(k)$ с использованием представления решений и функции приспособленности, описанных в п. 2.1.2. Пусть здесь так же, как в п. 2.1.2, используется оператор мутации из КГА с вероятностью мутации P_{m}.

Допустим, что всякие три гена, соответствующие одной трехвершинной клике, составляют один блок, а свойство $\mathbf{K}$ означает оптимальное по-

крытие блока. Тогда, рассмотрев всевозможные варианты образования избыточных покрытий трехвершинной клики, нетрудно убедиться в том, что $\tilde{r} = 1 - P_{\mathrm{m}}^3 - (1 - P_{\mathrm{m}})^3, r = 1 - P_{\mathrm{m}}(1 - P_{\mathrm{m}})^2 - P_{\mathrm{m}}^2(1 - P_{\mathrm{m}})$. Заметим, что неравенство $r \geq \tilde{r}$ выполняется при любой вероятности мутации P_{m}, следовательно, оператор мутации является монотонным.

Пример 2. Пусть функция приспособленности $\Phi(\xi)$ является *строго монотонной* [113], т. е.

$$\Phi(\xi) > \Phi(\xi') \text{ для любых } \xi, \xi' \in \mathcal{B}, \text{ таких что } \delta(\xi^*, \xi) < \delta(\xi^*, \xi'),$$

где $\delta(\cdot, \cdot)$ – расстояние Хэмминга, ξ^* – единственный максимум функции Φ на $\mathcal{B}$. Обозначим через $\Phi(\mathcal{B})$ множество всех значений приспособленности, т. е. $\Phi(\mathcal{B}) = \{\Phi(\xi) : \xi \in \mathcal{B}\}$.

Утверждение 2.9. *Пусть* Mut^* *– оператор мутации КГА,* $P_{\mathrm{m}} < 1/2$, $\Phi(\xi)$ *достигает максимума на* $\mathcal{B}$ *в единственной точке* ξ^*, *и* $|\Phi(\mathcal{B})| = l + 1$. *Тогда для монотонности* Mut^* *относительно* $\{\Phi_j : j = 0, \ldots, d\} = \Phi(\mathcal{B})$ *необходимо и достаточно, чтобы* Φ *была строго монотонной.*

Доказательство. Если Φ строго монотонна, ее значения строго возрастают с ростом числа битов генотипа, совпадающих с ξ^*. Будем рассматривать каждый из таких битов, как блок, обладающий свойством **K**. Тогда по утверждению 2.8, при $\tilde{r} = (1 - r) = P_{\mathrm{m}} < 0.5$, рассматриваемый оператор мутации является монотонным.

Для доказательства необходимости заметим, что имеется единственный генотип ξ^*, для которого $\Phi(\xi^*) = \Phi_l$, причем $l = d$. Из монотонности Mut^* следует, что для любого i, $i = 0, ..., d$, вероятности $\mathbf{P}\{\Phi(\mathrm{Mut}^*(\xi)) \geq \Phi_j\}$, $j = 1, ..., d$, не зависят от выбора вектора ξ с приспособленностью Φ_i. Обозначим эти вероятности, как и ранее, через μ_{ij}.

Покажем, что не существует такого i, $0 \leq i < d$, что $\mu_{id} = \mu_{i+1,d}$. Действительно, заметим, что имеется $d+1$ различное значение вероятности

$$\mathbf{P}\{\mathrm{Mut}^*(\xi) = \xi^*\} = P_{\mathrm{m}}^{\delta(\xi,\xi^*)}(1 - P_{\mathrm{m}})^{d-\delta(\xi,\xi^*)}. \tag{2.25}$$

Таким образом, если предположить, что для некоторого $i,\ 0 \le i < d$, выполняется $\mu_{id} = \mu_{i+1,d}$, то не все из $d+1$ возможных значений $\mathbf{P}\{\mathrm{Mut}^*(\xi) = \xi^*\}$ будут представлены среди вероятностей $\mu_{0d}, \mu_{1d}, ..., \mu_{dd}$. Это бы означало, что для некоторого $j,\ 0 \le j < d$, существуют генотипы ξ и ξ' с приспособленностью Φ_j и $\mu_{jd} = \mathbf{P}\{\mathrm{Mut}^*(\xi) = \xi^*\} \neq \mathbf{P}\{\mathrm{Mut}^*(\xi') = \xi^*\} = \mu_{jd}$, что невозможно. Следовательно, $\mu_{id} < \mu_{i+1,d}$ ввиду монотонности Mut^*. Кроме того, все генотипы с приспособленностью Φ_i равноудалены в метрике Хэмминга от ξ^*.

Из (2.25) заключаем, что для любого $i,\ i = 0, ..., d$, генотип ξ будет иметь приспособленность Φ_i тогда и только тогда, когда $\delta(\xi, \xi^*) = d - i$. Следовательно, функция $\Phi(\xi)$ является строго монотонной. $\square$

Предположение о единственности оптимального генотипа не может быть опущено в формулировке утверждения 2.9. Это связано с тем, что за пределами класса функций строго монотонных по уровням имеются функции с несколькими оптимумами, которые удовлетворяют всем прочим условиям данного утверждения. В случае $n = 3$, например, такой функцией является $\Phi(\xi) = \min\{k(\xi), 7 - k(\xi)\}$, где $k(\xi) = \xi_1 + 2\xi_2 + 4\xi_3$.

Кумулятивные вероятности перехода для операторов мутации из рассмотренных примеров 1 и 2 могут быть найдены по формуле (2.24).

Пример 3. В данном примере демонстрируется возможность использования оценок α_{ij} в случае, когда оператор мутации не является ступенчатым относительно рассматриваемого набора линий уровня. Рассмотрим семейство невзвешенных задач наименьшего покрытия множества, предложенных Э. Балашем в [118]. Обозначим через $N_i = \{j : i \in M_j\}$ множество покрывающих подмножеств, содержащих элемент $i,\ i = 1, \ldots, m$. В семействе задач Балаша $B(n, p)$ базовое множество имеет мощность $m = C_n^{p-1}$, а совокупность подмножеств $\{N_1, N_2, ..., N_m\}$ состоит из всех $(n - p + 1)$-

элементных подмножеств множества U. В таком случае $J \subseteq U$ – наименьшее покрытие тогда и только тогда, когда $|J| = p$. Как показано в [91], задачи данного семейства являются трудными для метода перебора L-классов и метода ветвей и границ Лэнд и Дойг [93]. В частности, при n четном и $p = n/2$ мощность L-накрытия растет экспоненциально с увеличением числа переменных. Проведем анализ ГА с турнирной селекцией в частном случае $p = n/2$.

Пусть используется *двоичное представление решений* [127], где $\ell = n$ и каждый ген $\xi_j \in \{0, 1\}, j \in U$, служит индикатором вхождения j в покрытие, т. е. $\Psi(\xi) = \{j \in U : \xi_j = 1\}$. Если $\Psi(\xi)$ – покрытие, то полагаем $\Phi(\xi) = n - |\Psi(\xi)| + 1$, иначе вводится штраф: $\Phi(\xi) = P(\xi)$, где $P(\xi) < 1$ – строго убывающая функция от числа непокрытых элементов из M.

Рассмотрим оператор точечной мутации (см. п. 1.2.1) с вероятностью мутации $P' > 0$. Пусть $d = n/2 = p$ и пороговые значения $\Phi_0, \Phi_1, ..., \Phi_p$ равны приспособленностям генотипов, содержащих $0, 1, ..., p$ единичных генов соответственно. Таким образом, при $\Phi(\xi) = \Phi_i,\ i = 0, \ldots, d-1$, генотип кодирует недопустимое решение, а генотипы, соответствующие допустимым решениям, принадлежат H_d.

Применимы следующие нижние оценки вероятностей: $\alpha_{ij} = 1$ при всех $i = 1, \ldots, d,\ j = 0, \ldots, i-1$; $\alpha_{i,i+1} = P'(n-i)/n$ при $i = 0, \ldots, d-1$;

$$\alpha_{ii} = \begin{cases} 1 - P' + \alpha_{i,i+1} & \text{при } \ i = 1, \ldots, d-1; \\ 1 - P'/2 & \text{при } \ i = d; \end{cases}$$

$\alpha_{i,j} = 0,\ i = 0, \ldots, d-2,\ j = i+2, \ldots, d$. При $i < d$ данные оценки совпадают с соответствующими кумулятивными вероятностями перехода. Условие монотонности для матрицы $\mathbf{A}$ выполняется при $P' \leq n/(n+1)$.

С целью применения утверждения 2.5 рассмотрим матричную норму $||\mathbf{W}||_\infty = \max_{i=1,\ldots,d} \sum_{j=1}^{d} |w_{ij}|$, подчиненную векторной норме $||\mathbf{z}||_1 = \sum_{j=1}^{d} |z_j|$ при умножении вектора-строки на матрицу слева. Для матрицы $\mathbf{W}$, отвечающей указанному выше набору нижних оценок α_{ij}, име-

ем $||\mathbf{W}||_\infty = 1 - 2P'/n$, т. е. условие $||\mathbf{W}^t||_\infty \stackrel{t\to\infty}{\longrightarrow} 0$ выполнено при $P' > 0$.

Найдем предельный вектор популяции $\mathbf{v} = \mathbf{a}(\mathbf{I} - \mathbf{W})^{-1}$ при $t \to \infty$. Для этого достаточно решить систему уравнений:

$$-v_{i-1}\frac{n-i+1}{n} + v_i\frac{n+1}{n} - v_{i+1}\frac{i}{n} = 0, \quad i = 2, \ldots, \frac{n}{2} - 1, \tag{2.26}$$

$$v_1\frac{n+1}{n} - v_2\frac{1}{n} = 1, \qquad -v_{n/2-1}\frac{n+2}{2n} + v_{n/2}\frac{n-1}{n} = 0. \tag{2.27}$$

Используя формулу для стационарного распределения в модели случайного блуждания П. Эренфеста и Т. Афанасьевой-Эренфест (см., например, [100], гл. 15, § 6), положим

$$v_i = \sum_{k=i}^{n/2} \binom{n}{k} \frac{1}{2^{n-1}}, \quad i = 1, \ldots, \frac{n}{2}. \tag{2.28}$$

Выполнение условий (2.26), (2.27) проверяется подстановкой.

Пусть $\mathbf{e} = (1, \ldots, 1)$. По свойствам используемых здесь норм $\mathbf{v}\mathbf{W}^t \leq \mathbf{e}||\mathbf{v}\mathbf{W}^t||_1 \leq \mathbf{e}||\mathbf{v}||_1 \cdot ||\mathbf{W}||_\infty^t \leq d||\mathbf{W}||_\infty^t\mathbf{e}$. По утверждению 2.5 для любого t

$$\mathbf{E}[\mathbf{z}^{(t)}] \geq \mathbf{E}[\mathbf{z}^{(0)}]\mathbf{W}^t + \mathbf{a}(\mathbf{I} - \mathbf{W})^{-1}(\mathbf{I} - \mathbf{W}^t) \geq \mathbf{v} - d||\mathbf{W}||_\infty^t\mathbf{e}.$$

Отсюда при $P' = n/(n+1)$ средняя доля генотипов, кодирующих допустимые решения, оценивается снизу величиной $v_d - d\left(\frac{n-1}{n+1}\right)^t$. Из (2.28) с использованием формулы Стирлинга получаем $v_d = \frac{\binom{n}{n/2}}{2^{n-1}} = \Omega(n^{-\frac{1}{2}})$. Заметим, что при достаточно большом номере итерации $t \geq \frac{n+1}{2}\ln\frac{n}{v_d}$,

$$\frac{v_d}{n} \geq \left(\frac{1}{e}\right)^{\frac{2t}{n+1}} \geq \left(\left(1 - \frac{2}{n+1}\right)^{\frac{n+1}{2}}\right)^{\frac{2t}{n+1}} = \left(\frac{n-1}{n+1}\right)^t,$$

а значит, $\frac{n}{2}\left(\frac{n-1}{n+1}\right)^t \leq \frac{v_d}{2}$, и $\mathbf{E}[z_d^{(t)}] \geq \frac{v_d}{2}$.

Пусть ГА начинает работу с популяции, состоящей только из нулевых генотипов. Поскольку рассматриваемый оператор мутации всякий раз изменяет число покрывающих подмножеств не более чем на 1, наличие хотя бы одного генотипа из H_d в текущей популяции означает, что за время

работы алгоритма оптимальное решение задачи из $B(n, n/2)$ было получено хотя бы раз. Таким образом, ввиду утверждения 2.2 после $\Omega(n \ln n)$ итераций ГА с $P' = n/(n+1)$ вероятность достижения оптимума задачи из семейства $B(n, n/2)$ есть $\Omega(n^{-0.5})$.

Заключение

Предложенная модель может применяться для исследования ГА при различных функциях приспособленности и операторах мутации. Полученные оценки математического ожидания доли генотипов с высоким уровнем приспособленности позволяют судить о динамике численности таких генотипов в популяции ГА. Нижние оценки оказываются достижимыми в случае $s = 1$, что соответствует равновероятной селекции родительских особей. Задача построения нижних оценок, учитывающих размер турнира s, остается открытой. Перспективным подходом к этой проблеме представляется метод П. Витани [288].

Выделенный здесь случай монотонного оператора мутации характеризует ситуацию, в которой при $N = 1$ становятся точными нижние оценки, а при $N \to \infty$ – асимптотически точными верхние оценки на математическое ожидание доли генотипов с высоким уровнем приспособленности. Как следует из утверждения 2.9, для оператора мутации классического ГА условия монотонности во многих случаях эквивалентны строгой монотонности функции приспособленности.

Теорема 2.2 показывает, что при достаточно большой численности популяции и монотонном операторе мутации увеличение размера турнира в среднем ведет к увеличению численности особей с достаточно высокой приспособленностью.

Открытым является вопрос о существовании эффективного алгоритма генерации генотипов с распределением особей на заданной итерации ГА с кроссинговером и бесконечной численностью популяции [114, 260].

2.3. Сравнение эволюционных алгоритмов с эволюционной стратегией (1+1)-ES

Основным результатом данного раздела является теорема о сравнении эволюционных алгоритмов с эволюционной стратегией (1+1)-ES, полученная в п. 2.3.1. Теорема дает необходимые и достаточные условия, при которых (1+1)-ES оказывается «наилучшим» методом в классе эволюционных алгоритмов с точки зрения распределения вероятностей приспособленности лучшего полученного генотипа на любой заданной итерации. Несколько простых примеров иллюстрируют случаи, когда упомянутые выше условия выполняются. Как показано в п. 2.3.1, в условиях применимости указанной теоремы (1+1)-ES не уступает никакому другому ЭА в терминах математического ожидания приспособленности лучшего полученного генотипа на любой заданной итерации, а также по среднему времени поиска оптимального генотипа.

Для удобства исследования в данном разделе мы будем иначе определять класс эволюционных алгоритмов, чем в п. 1.2.2. Основное отличие будет состоять в том, что здесь в явном виде допускается возможность использования неограниченной памяти о «предыстории» работы алгоритма, в то время как в определении ЭА из п. 1.2.2 для этого требуется искусственно увеличивать численность популяции в процессе вычислений.

Будем называть *эволюционным алгоритмом с неограниченной памятью* следующий рандомизированный алгоритм: начальная популяция $\xi^{1,0}, \ldots, \xi^{N,0}$ строится некоторым случайным или детерминированным способом. Пусть t, как и ранее, обозначает номер итерации. Последовательность генотипов, построенных в ЭА с неограниченной памятью к началу итерации t, обозначим через σ^{t-1}, а множество всех входящих в нее элементов – через $\mathrm{A}^{(t-1)}$. На каждой итерации t, $t = 1, 2, \ldots$, вычисляются N'' генотипов потомков $\xi^{1,t}, \ldots, \xi^{N''t}$ посредством применения оператора вос-

произведения $\mathrm{Rep}(\eta^1, \ldots, \eta^{N'})$, где $\eta^1, \ldots, \eta^{N'}$ – некоторые из ранее построенных особей. Эти родительские особи $\eta^1, \ldots, \eta^{N'}$ на итерации t выбираются с помощью рандомизированного оператора *селекции с неограниченной памятью* $\mathrm{Sel}_\infty : \mathcal{B}^{N+N''(t-1)} \to \mathcal{B}^{N'}$, так что $\mathrm{Sel}_\infty(\sigma^{t-1}) \subseteq \mathrm{A}^{(t-1)}$.

Работа ЭА с неограниченной памятью представляет собой итерации следующей композиции случайных отображений:

$$X^t = \mathrm{Rep}(\mathrm{Sel}_\infty(X^0, \ldots, X^{t-1})), \quad t = 1, 2, \ldots.$$

Алгоритм, удовлетворяющий этому определению, обозначим через EA.

2.3.1. Теорема о сравнении эволюционных алгоритмов с эволюционной стратегией (1+1)-ES

Пусть текущий генотип на итерации t, $t = 0, 1, \ldots$, эволюционной стратегии (1+1)-ES, как и ранее, обозначается через $\zeta^{(t)}$.

Будем обозначать максимум функции приспособленности на какой-либо последовательности генотипов $\sigma = (\xi_1, \ldots, \xi_k)$ через $\check{\Phi}(\sigma)$, т. е. $\check{\Phi}(\sigma) = \max_{i=1,\ldots,k} \Phi(\xi_i)$.

Сравнение оператора воспроизведения алгоритма EA и оператора мутации эволюционной стратегии (1+1)-ES будет осуществляться на основе следующего определения.

Определение 2.3. *Оператор воспроизведения* Rep *доминируется оператором мутации* Mut, *если для любой N'-элементной последовательности генотипов $X' = (\xi^1, \ldots, \xi^{N'})$ и любого $\eta \in \mathcal{B}$, таких что $\Phi(\eta) \geq \check{\Phi}(X')$, выполняется следующее условие при всех $\phi > \Phi(\eta)$:*

$$\mathbf{P}\left\{\Phi(\mathrm{Mut}(\eta)) \geq \phi\right\} \geq \mathbf{P}\left\{\check{\Phi}(\mathrm{Rep}(X')) \geq \phi\right\}. \tag{2.29}$$

Данное определение может интерпретироваться как требование того, чтобы вероятность получения генотипа достаточно высокой приспособленности посредством оператора мутации Mut была бы не меньше, чем вероятность такого же события для оператора рекомбинации Rep, если входной

генотип оператора Mut не уступает ни одному из родительских генотипов на входе Rep.

В качестве простого примера доминирования рассмотрим случай, когда $\Phi \in \text{ONEMAX}^{**}$ и Rep – равномерный кроссинговер. Пусть оператор Mut действует при заранее известном оптимальном генотипе ξ^*: если ген ξ_i родительского решения совпадает с ξ_i^*, то $\text{Mut}(\xi)_i = \xi_i$, иначе ген i в решении-потомке выбирается равным 0 или 1 с одинаковыми вероятностями. Легко проверить, что такой оператор Mut доминирует Rep.

Следующая теорема показывает, что если начальный генотип (1+1)-ES не уступает по приспособленности лучшей особи начальной популяции EA, то на любой итерации распределение вероятностей текущего генотипа (1+1)-ES также не уступает распределению лучшего из найденных генотипов в EA тогда и только тогда, когда Mut доминирует Rep.

Сформулируем теорему о сравнении эволюционных алгоритмов с (1+1)-ES.

Теорема 2.3. *Для того чтобы к любому алгоритму EA из класса ЭА с неограниченной памятью при*

$$\Phi(\zeta^{(0)}) \geq \max_{i=1,\ldots,N} \Phi(\xi^{(i,0)}) \tag{2.30}$$

были применимы неравенства

$$\mathbf{P}\{\Phi(\zeta^{(t)}) \geq \phi\} \geq \mathbf{P}\{\check{\Phi}(\sigma^t) \geq \phi\}, \quad t \in \mathbb{Z}_+, \quad \phi \in \mathbb{R}, \tag{2.31}$$

необходимо и достаточно, чтобы оператор Rep *доминировался оператором* Mut.

Доказательство.

1. Достаточность. Пусть множество всевозможных значений функции приспособленности есть $\{\Phi_0, \ldots, \Phi_d\}$, где $\Phi_0 < \ldots < \Phi_d$. Пользуясь индукцией по t, предположим, что

$$\mathbf{P}\{\Phi(\zeta^{(t-1)}) \geq \Phi_j\} \geq \mathbf{P}\{\check{\Phi}(\sigma^{t-1}) \geq \Phi_j\} \text{для всех } j = 0, \ldots, d$$

(базис индукции при $t = 0$ следует из формулировки теоремы). На итерации t алгоритма EA имеем $\check{\Phi}(\sigma^t) = \max\{\Phi(\xi'), \check{\Phi}(\sigma^{t-1})\}$, где ξ' – генотип с наибольшей приспособленностью среди потомков, построенных оператором $\mathrm{Rep}(\eta^1, \ldots, \eta^{N'})$, при $\{\eta^1, \ldots, \eta^{N'}\} \subseteq \mathrm{A}^{(t-1)}$, выбранных оператором Sel_∞.

Зафиксируем произвольное j, $j \in \{0, \ldots, d\}$, и введем обозначения для каждого $i = 0, \ldots, j$:

$$p_i = \mathbf{P}\{\check{\Phi}(\sigma^{t-1}) \geq \Phi_i\}, \quad v_{ij} = \mathbf{P}\{\Phi(\xi') \geq \Phi_j \mid \check{\Phi}(\sigma^{t-1}) = \Phi_i\}.$$

Тогда

$$\mathbf{P}\{\check{\Phi}(\sigma^t) \geq \Phi_j\} = p_j + \mathbf{P}\{\Phi(\xi') \geq \Phi_j, \check{\Phi}(\sigma^{t-1}) < \Phi_j\} = \\ = p_j + \sum_{i=0}^{j-1} v_{ij}(p_i - p_{i+1}) \tag{2.32}$$

по формуле полной вероятности.

Аналогичные рассуждения для (1+1)-ES дают

$$\mathbf{P}\{\Phi(\zeta^{(t)}) \geq \Phi_j\} = q_j + \sum_{i=0}^{j-1} w_{ij}(q_i - q_{i+1}), \tag{2.33}$$

где $q_i = \mathbf{P}\{\Phi(\zeta^{(t-1)}) \geq \Phi_i\}$ и $w_{ij} = \mathbf{P}\{\Phi(\zeta') \geq \Phi_j \mid \Phi(\zeta^{(t-1)}) = \Phi_i\}$, $i = 0, \ldots, j$, при $\zeta' = \mathrm{Mut}(\zeta^{(t-1)})$.

Условие доминирования означает, что для любых i, k, $i \leq k$, $k < j$, выполняется $v_{ij} \leq w_{kj}$. Пусть $\hat{v}_{ij} = \max\limits_{l=0,\ldots,i} v_{lj}$ для каждого $i = 0, \ldots, j-1$. Тогда $\hat{v}_{i-1,j} \leq \hat{v}_{i,j}$ и

$$v_{ij} \leq \hat{v}_{ij} \leq w_{ij}, \quad i = 0, \ldots, j-1.$$

Используя эти свойства, выражения (2.32) и (2.33), а также преобразование Абеля, получаем:

$$\mathbf{P}\{\check{\Phi}(\sigma^t) \geq \Phi_j\} = p_j + \sum_{i=0}^{j-1} v_{ij}(p_i - p_{i+1}) \leq p_j + \sum_{i=0}^{j-1} \hat{v}_{ij}(p_i - p_{i+1}) =$$

$$= \hat{v}_{0j} + \sum_{i=1}^{j-1} (\hat{v}_{ij} - \hat{v}_{i-1,j})p_i + (1 - \hat{v}_{j-1,j})p_j \leq$$

$$\leq \hat{v}_{0j} + \sum_{i=1}^{j-1}(\hat{v}_{ij} - \hat{v}_{i-1,j})q_i + (1 - \hat{v}_{j-1,j})q_j =$$

$$= q_j + \sum_{i=0}^{j-1}\hat{v}_{ij}(q_i - q_{i+1}) \leq q_j + \sum_{i=0}^{j-1} w_{ij}(q_i - q_{i+1}) = \mathbf{P}\{\Phi(\zeta^{(t)}) \geq \Phi_j\}.$$

2. Необходимость. Рассмотрим алгоритм EA, где $N = N'$, а оператор селекции таков, что всякий раз выбираются N' последних построенных EA генотипов из последовательности σ^t. Согласно условиям теоремы при любой N'-элементной последовательности генотипов $X^0 = (\xi^{(1,0)}, \ldots, \xi^{(N',0)})$ и любом $\zeta^{(0)} \in \mathcal{B}$, удовлетворяющих условию (2.30), т. е. таких, что $\Phi(\zeta^{(0)}) \geq \check{\Phi}(X^0)$, для всех $\phi > \Phi(\zeta^{(0)})$ имеем:

$$\mathbf{P}\left\{\Phi(\mathrm{Mut}(\zeta^{(0)})) \geq \phi\right\} = \mathbf{P}\{\Phi(\zeta^{(1)}) \geq \phi\} \geq$$

$$\geq \mathbf{P}\{\check{\Phi}(\sigma^1) \geq \phi\} = \mathbf{P}\left\{\check{\Phi}(\mathrm{Rep}(X^0)) \geq \phi\right\},$$

т. е. оператор Rep доминируется оператором Mut. □

Обобщение данной теоремы на случай ЭА, оперирующих генотипами с вещественными значениями генов, может быть получено с помощью результата из [137].

2.3.2. Монотонные операторы воспроизведения

В некоторых случаях для оператора Rep существует достаточно простой способ построения оператора мутации, доминирующего его. Определим оператор мутации, соответствующий оператору Rep следующим образом

$$\mathrm{Mut}_{\mathrm{Rep}}(\zeta) = \mathrm{argmax}\ (\Phi(\zeta), \Phi(\xi^1), \ldots, \Phi(\xi^{N''})),$$

где $(\xi^1, \ldots, \xi^{N''}) = \mathrm{Rep}(\zeta, \ldots, \zeta)$. Таким образом, в $\mathrm{Mut}_{\mathrm{Rep}}$ сначала оператор воспроизведения Rep применяется к набору идентичных родительских генотипов, и после этого результат выбирается как генотип с наибольшей

приспособленностью среди генотипов, имеющихся на входе и выходе Rep. В случае, если при этом обнаружится несколько генотипов с одинаковой приспособленностью, будем предполагать, что $\mathrm{Mut}_{\mathrm{Rep}}(\zeta)$ выбирается среди них равновероятно. Введем следующее определение, обобщающее понятие монотонного оператора мутации из раздела 2.2.

Определение 2.4. *Оператор воспроизведения* Rep *называется монотонным, если для любых двух последовательностей из N' генотипов $\xi^1, \ldots, \xi^{N'}$ и $\bar{\xi}^1, \ldots, \bar{\xi}^{N'}$, таких что*

$$\Phi(\xi^1) \leq \Phi(\bar{\xi}^1), \ldots, \Phi(\xi^{N'}) \leq \Phi(\bar{\xi}^{N'}), \tag{2.34}$$

следующие условия выполняются для всех ϕ:

$$\mathbf{P}\left\{\max_{i=1,\ldots,N''} \Phi(\bar{\eta}^i) \geq \phi\right\} \geq \mathbf{P}\left\{\max_{i=1,\ldots,N''} \Phi(\eta^i) \geq \phi\right\}, \tag{2.35}$$

где $(\eta^1, \ldots, \eta^{N''}) = \mathrm{Rep}(\xi^1, \ldots, \xi^{N'})$ и $(\bar{\eta}^1, \ldots, \bar{\eta}^{N''}) = \mathrm{Rep}(\bar{\xi}^1, \ldots, \bar{\xi}^{N'})$.

Условие монотонности оператора воспроизведения означает, что замена родительских генотипов на генотипы большей или равной приспособленности не приводит к снижению шансов получения достаточно приспособленных потомков на его выходе.

Заметим, что если условия (2.34) выполнены как равенства для родительских генотипов $\xi^1, \ldots, \xi^{N'}$ и $\bar{\xi}^1, \ldots, \bar{\xi}^{N'}$, то ввиду (2.35) распределения вероятностей приспособленности лучшего потомка для $\mathrm{Rep}(\xi^1, \ldots, \xi^{N'})$ и $\mathrm{Rep}(\bar{\xi}^1, \ldots, \bar{\xi}^{N'})$ должны совпадать.

Если оператор Rep является монотонным, то $\mathrm{Mut}_{\mathrm{Rep}}$ доминирует Rep по построению, в таком случае алгоритм EA с оператором воспроизведения Rep может сравниваться по теореме 2.3 с (1+1)-ES, где используется оператор мутации $\mathrm{Mut}_{\mathrm{Rep}}$. Следующее следствие непосредственно вытекает из теоремы 2.3.

Следствие 2.2. *Пусть в EA используется монотонный оператор воспроизведения* Rep*, и оператор* $\mathrm{Mut}_{\mathrm{Rep}}$ *используется в (1+1)-ES. Пусть,*

кроме того, (1+1)-ES начинает работу с генотипа $\zeta^{(0)}$*, такого что* $\Phi(\zeta^{(0)}) \geq \max\limits_{i=1,\ldots,N} \Phi(\xi^{(i,0)})$*. Тогда для всех* $t \geq 0$ *и всех* $\phi \in \mathbb{R}$ *выполняется*

$$\mathbf{P}\{\Phi(\zeta^{(t)}) \geq \phi\} \geq \mathbf{P}\{\check{\Phi}(\sigma^t) \geq \phi\}. \tag{2.36}$$

Частным случаем монотонного оператора воспроизведения при $N' = N'' = 1$ является монотонный оператор мутации. Непосредственно из определений следует, что монотонный оператор мутации доминирует сам себя, и по следствию 2.2 в таком случае (1+1)-ES является «наилучшим» методом в классе эволюционных алгоритмов с неограниченной памятью. Примеры двух монотонных операторов мутации для задач комбинаторной оптимизации уже были рассмотрены в п. 2.2.2.

Приведем простой пример монотонного оператора воспроизведения, где $N' = 2, N'' = 1$. Пусть $\Phi \in \text{ONEMAX}^{**}$ и оператор Rep построен на основе равномерного кроссинговера, однако перед применением равномерного кроссинговера к генам одного из двух родительских генотипов применяется перестановка, выбранная равновероятно. В качестве результата воспроизведения используется первый из двух генотипов, полученных в результате равномерного кроссинговера.

Нетрудно видеть, что в таком случае Rep является монотонным, как следует из утверждения 2.8. Заметим, что без использования случайной перестановки такой оператор не был бы монотонным (например, для функции $\Phi(\xi) = \sum_{i=1}^{l} \xi_i$ условие (2.35) нарушается на родительских генотипах $\xi^1 = (1,0), \xi^2 = (0,1)$ $\bar{\xi}^1 = (1,0), \bar{\xi}^2 = (1,0)$ при $\phi = 2$).

Другие примеры монотонных операторов воспроизведения в случаях, когда гены принимают вещественные значения, приводятся в [137].

Все упомянутые выше примеры относятся к задачам регулярной структуры. На практике, однако, задачи комбинаторной оптимизации, как правило, не имеют такой структуры и свойство монотонности оператора воспроизведения выполняется редко. В частности, при наличии локаль-

ных оптимумов функции приспособленности, не являющихся глобальными, оператор точечной мутации (см. п. 1.2.1) не может быть монотонным. Как показывает утверждение 2.9, при достаточно общих предположениях то же верно и для оператора мутации классического ГА.

В ситуациях, когда условия монотонности не выполняются, (1+1)-ES может уступать по эффективности другим известным ЭА. Пример ситуации, когда (1+1)-ES существенно уступает ГА, приводится в [219]. В то же время известны серии задач о наибольшем независимом множестве вершин графа [139], в которых при «малом» в некотором смысле нарушении монотонности преимущество сохраняется за (1+1)-ES.

2.3.3. Среднее время достижения оптимума и средняя приспособленность на заданной итерации

Рассмотрим среднее число обращений к оператору воспроизведения до достижения множества генотипов требуемого качества. Особый интерес представляет множество генотипов, отображаемых функцией Ψ в оптимальные решения задачи комбинаторной оптимизации. Вместе с тем будем рассматривать и математическое ожидание приспособленности рекордного решения, получаемого к заданной итерации ЭА с неограниченной памятью.

Обозначим через $t_\phi^{(1+1)}$ среднее число итераций (1+1)-ES до получения генотипа с приспособленностью не ниже ϕ. Среднее число итераций ЭА с неограниченной памятью до получения генотипа с приспособленностью не ниже ϕ обозначим через t_ϕ^{EA}.

Следствие 2.3. *Пусть оператор* Rep *доминируется оператором* Mut *и*

$$\Phi(\zeta^{(0)}) \geq \max_{i=1,\ldots,N} \Phi(\xi^{(i,0)}), \tag{2.37}$$

тогда:

(i) всех $t \geq 0$ *выполняется*

$$\mathbf{E}[\Phi(\zeta^{(t)})] \geq \mathbf{E}[\check{\Phi}(\sigma^t)];$$

(ii) если t_ϕ^{EA} *конечно, то* $t_\phi^{EA} \geq t_\phi^{(1+1)}$.

Доказательство. (i) Пусть $F_1(\phi)$ и $F_2(\phi)$ обозначают функции распределения случайных величин $\Phi(\zeta^{(t)})$ и $\check{\Phi}(\sigma^t)$ соответственно. Тогда из теоремы 2.3 следует неравенство $F_1(\phi) \leq F_2(\phi)$ для любого $\phi \in \mathbb{R}$. Таким образом, по свойствам математического ожидания (см., например, [17, 25])

$$\mathbf{E}[\Phi(\zeta^{(t)})] = \int_0^\infty (1 - F_1(\phi))d\phi \geq \int_0^\infty (1 - F_2(\phi))d\phi = \mathbf{E}[\check{\Phi}(\sigma^t)]. \quad (2.38)$$

(ii) Аналогично неравенству (2.38) при конечном $\mathbf{E}[t]$ имеем

$$t_\phi^{EA} = \sum_{k=0}^{\infty}(1 - \mathbf{P}\{\check{\Phi}(\sigma^k) \geq \phi\})$$

и

$$t_\phi^{(1+1)} = \sum_{k=0}^{\infty}(1 - \mathbf{P}\{\Phi(\zeta^{(k)}) \geq \phi\}).$$

Применение теоремы 2.3 завершает доказательство. □

Теорема 2.3 и следствие 2.3 показывают, что если требуется сделать выбор между ЭА и эволюционной стратегией (1+1)-ES, то в условиях применимости этих результатов предпочтение следует отдать (1+1)-ES.

Заключение

В данном разделе проведено сравнение эволюционной стратегии (1+1)-ES с другими ЭА. В частности, показано, что если выполняется условие доминирования, то данная эволюционная стратегия является «наилучшим» ЭА с точки зрения вероятности получения решений требуемого качества к любой заданной итерации. В условиях доминирования (1+1)-ES является предпочтительным и с точки зрения математического ожидания приспособленности рекордного решения на любой итерации, а также по математическому ожиданию числа итераций до получения оптимума.

Доказательства теоремы 2.3 (в части достаточности), а также ее следствий 2.2 и 2.3 получены автором совместно с П.А. Борисовским.

2.4. Статистические оценки числа локальных оптимумов

Исследование эволюционных алгоритмов зачастую приводит к необходимости анализа структуры множества локальных оптимумов и, в частности, оценке их количества. Как отмечалось в п. 1.2.3, во многих ЭА перед добавлением в популяцию каждый новый генотип проходит процедуру локальной оптимизации. При использовании такого подхода возникает вопрос о мощности множества всех генотипов, получаемых в результате локальной оптимизации, так как мощность этого множества в некотором смысле характеризует сложность решаемой задачи для ЭА. Кроме того, существует большое число эвристических методов решения задач комбинаторной оптимизации, в которых локальный поиск используется многократно с различными начальными точками (см., например, [74, 274, 295]). Для успешного применения таких методов большое значение имеет удачный выбор системы окрестностей. Сложность индивидуальной задачи с заданной системой окрестностей характеризуется целым рядом параметров, один из которых – число локальных оптимумов. Наконец, в работе [291] М. Возом и А. Райтом выдвинута гипотеза, из которой вытекает существование тесной связи между устойчивыми неподвижными точками динамической системы, описывающей КГА при $N \to \infty$, и локальными оптимумами решаемой задачи.

В настоящем разделе предлагаются и обосновываются статистические методы построения доверительных интервалов для числа локальных оптимумов на основе результатов многократного выполнения локального поиска. Методы предназначены для исследования индивидуальных задач комбинаторной оптимизации, в которых большая размерность пространства решений затрудняет полный перебор всех его элементов.

Обозначим число локальных оптимумов задачи через ν. Из литературы известно несколько подходов к оцениванию величины ν на основе результатов многократного выполнения локального поиска из случайно вы-

бранных начальных точек. Один из подходов основывается на упрощающем предположении о том, что все локальные оптимумы имеют равные вероятности попадания в них (предположение *изотропности*) [270]. Другие подходы состоят в подборе параметров некоторого распределения вероятностей [189, 271] для описания вероятности попадания в различные локальные оптимумы или использовании таких непараметрических методов, как бутстреп-метод или метод складного ножа [180, 271, 272]. Некоторые из этих подходов основываются на методах, разработанных в статистике и экологии для оценивания численности популяций (см., например, [259, 264]).

Рассмотрим сначала достаточно общую ситуацию, когда при любом значении представляющего интерес неизвестного параметра ν функция распределения наблюдаемой целочисленной случайной величины T известна. Пусть $\mathbf{F}(t;\nu) = \mathbf{P}\{T \le t\}$ – функция распределения величины T при указанном значении ν. Наблюдаемое значение величины T, полученное в фактически проведенном эксперименте, будем обозначать через τ, $\tau \in \mathbb{N}$.

Если упрощенно считать, что ν может принимать любое вещественное значение и $\mathbf{F}(t;\nu)$ непрерывно зависит от ν, то при заданных вероятностях α_1, α_2, таких что $\alpha_1 + \alpha_2 < 1$, доверительный интервал $[\nu_1(\tau), \nu_2(\tau)]$ уровня $(1 - \alpha_1 - \alpha_2) \cdot 100\%$ может быть построен решением уравнений

$$1 - \mathbf{F}(\tau;\nu_1) = \alpha_1, \quad \mathbf{F}(\tau;\nu_2) = \alpha_2 \tag{2.39}$$

относительно неизвестных ν_1 и ν_2 (см., например, [94, 242]).

Поскольку в поставленной задаче параметр ν принимает только натуральные значения, необходимо соответственно адаптировать (2.39).

Утверждение 2.10. *Пусть*

$$\nu_1(\tau) = \min\{\nu' \in \mathbb{N} : 1 - \mathbf{F}(\tau - 1;\nu') \ge \alpha_1\},$$

$$\nu_2(\tau) = \max\{\nu' \in \mathbb{N} : \mathbf{F}(\tau;\nu') \ge \alpha_2\}$$

тогда $[\nu_1(\tau), \nu_2(\tau)]$ *является консервативным доверительным интервалом целочисленного параметра* ν *с уровнем значимости* $(1-\alpha_1-\alpha_2)\cdot 100\%$,

причем $\mathbf{P}\{\nu < \nu_1(T)\} \le \alpha_1$, $\mathbf{P}\{\nu_2(T) < \nu\} \le \alpha_2$.

Доказательство. Пусть для некоторого значения ν утверждение неверно. Тогда имеет место по крайней мере одно из двух неравенств: $\alpha_1 < \mathbf{P}\{\nu < \nu_1(\tau)\}$, $\alpha_2 < \mathbf{P}\{\nu_2(\tau) < \nu\}$.

В первом случае

$$\alpha_1 < \mathbf{P}\{\nu < \min(\nu' : 1 - \mathbf{F}(T-1;\nu') \ge \alpha_1)\} \le \mathbf{P}\{1 - \mathbf{F}(T-1;\nu) < \alpha_1\} =$$

$$= \mathbf{P}\{T - 1 \ge \min(\Theta : 1 - \mathbf{F}(\Theta;\nu) < \alpha_1)\} = \mathbf{P}\{T \ge \Theta_{\min}(\nu,\alpha_1)\},$$

где $\Theta_{\min}(\nu,\alpha_1) = \min\{\Theta' : 1 - \mathbf{F}(\Theta' - 1;\nu) < \alpha_1\}$. Однако

$$\mathbf{P}\{T \ge \Theta_{\min}(\nu,\alpha_1)\} = 1 - \mathbf{F}(\Theta_{\min}(\nu,\alpha_1) - 1;\nu) < \alpha_1 \qquad (2.40)$$

по определению $\Theta_{\min}$, что приводит к противоречию.

Во втором случае

$$\alpha_2 < \mathbf{P}\{\max\{\nu' : \mathbf{F}(T;\nu') \ge \alpha_2\} < \nu\} \le$$

$$\le \mathbf{P}\{\mathbf{F}(T;\nu) < \alpha_2\} = \mathbf{P}\{T < \theta_{\min}(\nu,\alpha_2)\},$$

где $\theta_{\min}(\nu,\alpha_2) = \min\{\theta : \mathbf{F}(\theta;\nu) \ge \alpha_2\}$. Однако $\mathbf{P}\{T < \theta_{\min}(\nu,\alpha_2)\} < \alpha_2$. □

Модифицируем предыдущее утверждение с целью нахождения консервативной нижней границы заданного доверительного уровня α в случае, когда распределение вероятностей случайной величины T зависит не только от ν, но и от дополнительного вектора параметров $\mathbf{y}$, принадлежащего некоторому множеству $\mathcal{Y}(\nu) \subseteq \mathbb{R}^{\nu}$. Обозначим в таком случае функцию распределения для T через $\mathbf{F}(t;\nu,\mathbf{y})$.

Утверждение 2.11. *Пусть*

$$\nu_1(\tau) = \min\left\{\nu' \in \mathbb{N} \ : \ 1 - \inf_{\mathbf{y}\in\mathcal{Y}(\nu')} \mathbf{F}(\tau - 1;\nu',\mathbf{y}) \ \ge \ \alpha\right\},$$

тогда $\mathbf{P}\{\nu < \nu_1(T)\} \le \alpha$.

Доказательство. Достаточно в доказательстве предыдущего утверждения заменить определение $\Theta_{\min}(\nu, \alpha)$ на

$$\Theta_{\min}(\nu, \alpha) = \min\left\{\Theta' \in \mathbb{N} \ : \ 1 - \inf_{\mathbf{y} \in \mathcal{Y}(\nu)} \mathbf{F}(\Theta' - 1; \nu, \mathbf{y}) \ < \ \alpha\right\},$$

и соответствующим образом модифицировать неравенство (2.40):

$$\mathbf{P}\{T \geq \Theta_{\min}(\nu, \alpha)\} = 1 - \mathbf{F}(\Theta_{\min}(\nu, \alpha) - 1; \nu, \mathbf{y}) \leq$$

$$\leq 1 - \inf\{\mathbf{F}(\Theta_{\min}(\nu, \alpha) - 1; \nu, \mathbf{y}) \ : \ \mathbf{y} \in \mathcal{Y}(\nu)\} < \alpha.$$

□

Если при любом t функция $\mathbf{F}(t; \nu)$ (функция $\mathbf{F}_{\mathcal{Y}}(t, \nu) = \inf_{\mathbf{y} \in \mathcal{Y}(\nu')} \mathbf{F}(\tau - 1; \nu', \mathbf{y})$) монотонна по ν, то доверительный интервал из утверждения 2.10 (утверждения 2.11) может быть вычислен с помощью метода дихотомии. Ниже будет показано, что для рассматриваемых в настоящем разделе оценок величины ν монотонность по ν имеет место.

2.4.1. Ожидание первого повторения

Пусть p_i – вероятность попадания локального поиска в локальный оптимум i, $i = 1, \ldots, \nu$, и пусть T – номер испытания локального поиска, при котором впервые получен ранее встречавшийся локальный оптимум. Функцию распределения случайной величины T, соответствующую вектору $\mathbf{p} = (p_1, ..., p_\nu)$, будем обозначать через $\mathbf{F}_{\mathbf{p}}(t; \nu)$. В частном случае *изотропного распределения*, когда вероятности попадания во все локальные оптимумы одинаковы, введем обозначение $\overline{\mathbf{p}} = (1/\nu, ..., 1/\nu)$.

Следующее утверждение показывает, что изотропное распределение является в некотором смысле экстремальным.

Утверждение 2.12. *Для любых* $t, \nu \in \mathbb{N}$, $t \geq 2$, *функция* $\mathbf{F}_{\mathbf{p}}(t; \nu)$ *достигает своего единственного минимума при* $\mathbf{p} = \overline{\mathbf{p}}$.

Доказательство. Пусть S – множество всех последовательностей длины t из элементов множества $\{1, ..., \nu\}$, не содержащих повторений, и

пусть $\sigma(i)$ – i-й элемент последовательности σ. Тогда

$$\mathbf{F}_{\mathbf{p}}(t;\nu) = 1 - \mathbf{P}\{T > t\} = 1 - \sum_{\sigma \in S} \prod_{i=1}^{t} \mathbf{p}_{\sigma(i)}.$$

$\mathbf{F}_{\mathbf{p}}(t;\nu)$ – непрерывная функция на компактном множестве $\Lambda = \{(p_1, p_2, ..., p_\nu) \geq 0 : \sum_{i=1}^{\nu} p_i = 1\}$, когда t фиксировано. Таким образом, существует распределение вероятностей попадания в локальные оптимумы, при котором достигается минимум функции $\mathbf{F}_{\mathbf{p}}(t;\nu)$ на Λ. Покажем, что никакое распределение $\mathbf{p}$, кроме $\overline{\mathbf{p}}$, таким свойством не обладает.

Предположим противное: пусть минимум достигается на векторе $\mathbf{p}$, где $p_i \neq p_j$ для некоторых i и j. Не теряя общности предположим, что $i = 1,\ j = 2,\ p_2 = \gamma p_1$ и $\gamma < 1$. Рассмотрим распределение вероятностей:

$$p'_1 = p'_2 = \frac{p_1 + p_2}{2},\ p'_3 = p_3,\ \ldots, p'_\nu = p_\nu,$$

и соответствующую ему функцию распределения $\mathbf{F}_{\mathbf{p}'}(t;\nu)$. Достаточно показать, что $\mathbf{F}_{\mathbf{p}'}(t;\nu) < \mathbf{F}_{\mathbf{p}}(t;\nu)$, так как это приведет к противоречию с оптимальностью $\mathbf{p}$. Обозначим через $S(j)$ подмножество S, состоящее из тех последовательностей, в которые входит элемент $j,\ j = 1, 2$. Тогда $\mathbf{F}_{\mathbf{p}}(t;\nu) = 1 - a(\mathbf{p}) - b(\mathbf{p}) - c(\mathbf{p}) - d(\mathbf{p})$, где

$$a(\mathbf{p}) = \sum_{\sigma \in S(1) \cap S(2)} \prod_{i=1}^{t} p_{\sigma(i)}, \qquad b(\mathbf{p}) = \sum_{\sigma \in S(1) \setminus S(2)} \prod_{i=1}^{t} p_{\sigma(i)},$$

$$c(\mathbf{p}) = \sum_{\sigma \in S(2) \setminus S(1)} \prod_{i=1}^{t} p_{\sigma(i)}, \qquad d(\mathbf{p}) = \sum_{\sigma \in S \setminus (S(1) \cup S(2))} \prod_{i=1}^{t} p_{\sigma(i)}.$$

Аналогично $\mathbf{F}_{\mathbf{p}'}(t;\nu) = 1 - a(\mathbf{p}') - b(\mathbf{p}') - c(\mathbf{p}') - d(\mathbf{p}')$ и $d(\mathbf{p}') = d(\mathbf{p})$. Заметим, что если элемент 2 во всех последовательностях из $S(2) \setminus S(1)$ заменить на 1, то будет получено множество последовательностей $S(1) \setminus S(2)$. Таким образом, с одной стороны,

$$\begin{aligned} b(\mathbf{p}) + c(\mathbf{p}) &= \sum_{\sigma \in S(1) \setminus S(2)} \prod_{i=1}^{t} p_{\sigma(i)} + \gamma \left(\sum_{\sigma \in S(1) \setminus S(2)} \prod_{i=1}^{t} p_{\sigma(i)} \right) \\ &= (1 + \gamma) p_1 \cdot \sum_{\sigma \in S(1) \setminus S(2)} \prod_{i : \sigma(i) \neq 1} p_{\sigma(i)}. \end{aligned}$$

С другой стороны, $p'_1 = p'_2 = (p_1 + p_2)/2$, поэтому

$$b(\mathbf{p}') + c(\mathbf{p}') = (p_1 + p_2) \sum_{\sigma \in S(1) \setminus S(2)} \prod_{i:\sigma(i) \neq 1} p_{\sigma(i)}$$

и $b(\mathbf{p}) + c(\mathbf{p}) = b(\mathbf{p}') + c(\mathbf{p}')$. Далее

$$a(\mathbf{p}) = \gamma p_1^2 \cdot \sum_{\sigma \in S(1) \cap S(2)} \prod_{i:\sigma(i) \neq 1,2}^{t} p_{\sigma(i)},$$

$$a(\mathbf{p}') = \left(\frac{p_1(1+\gamma)}{2} \right)^2 \sum_{\sigma \in S(1) \cap S(2)} \prod_{i:\sigma(i) \neq 1,2}^{t} p_{\sigma(i)}.$$

Однако $(1+\gamma)^2 > 4\gamma$, поэтому $a(\mathbf{p}') > a(\mathbf{p})$ и $\mathbf{F}_{\mathbf{p}'}(t;\nu) < \mathbf{F}_{\mathbf{p}}(t;\nu)$. □

Как следствие из доказанного утверждения вытекает свойство монотонности функции $\mathbf{F}_{\overline{\mathbf{p}}}(t;\nu)$.

Следствие 2.4. $\mathbf{F}_{\overline{\mathbf{p}}}(t;\nu)$ *является строго убывающей функцией от* ν.

Доказательство. Достаточно сравнить оптимальное по утверждению 2.12 значение $\mathbf{F}_{\overline{\mathbf{p}}}(t;\nu)$ с величиной $\mathbf{F}_{\mathbf{p}}(t;\nu)$, где $\mathbf{p} = (1/(\nu-1), 1/(\nu-1), \ldots, 1/(\nu-1), 0)$. □

Ввиду монотонности $\mathbf{F}_{\overline{\mathbf{p}}}(t;\nu)$ по ν в изотропном случае доверительный интервал для ν может быть вычислен методом дихотомии с использованием утверждения 2.10, при этом $\mathbf{F}(t;\nu) = 1 - \binom{\nu}{t} t!/\nu^t$. Консервативная нижняя граница ν_1 с заданным доверительным уровнем $100 \cdot (1-\alpha)\%$, справедливая при любом распределении $\mathbf{p}$, вытекает из утверждения 2.11. Далее будем обозначать эту границу через $L(t, 1-\alpha)$.

Если задача оптимизации имеет чрезвычайно большое число локальных оптимумов, а число испытаний локального поиска r оказалось недостаточным, чтобы встретить хотя бы один локальный оптимум более одного раза, то консервативная нижняя граница доверительного уровня $100(1-\alpha)\%$ для ν может быть вычислена в предположении о том, что следующее (гипотетическое) испытание приведет к первому повторению.

Оценим скорость роста оценки $L(r+1, 1-\alpha)$ как функции от r. Заметим, что согласно утверждению 2.11 $L(r+1, 1-\alpha)$ есть минимальное значение μ, при котором $\binom{\mu}{r} r!/\mu^r \geq \alpha$. Однако

$$\binom{\mu}{r}\frac{r!}{\mu^r} = \frac{\mu!}{(\mu-r)!\mu^r} = \exp\left\{\sum_{j=1}^{r} \ln\left(1 - \frac{r-j}{\mu}\right)\right\} \leq$$

$$\leq \exp\left\{-\sum_{j=1}^{r}\frac{r-j}{\mu}\right\} = \exp\left\{-\frac{r(r+1)}{2\mu}\right\} \leq \exp\left\{-\frac{r^2}{2\mu}\right\},$$

поэтому $L(r+1, 1-\alpha) \geq 0.5r^2/\ln(\alpha^{-1})$, и, к примеру, при $\alpha = 0.05$ имеем $0.5r^2/\ln(\alpha^{-1}) \approx 0.167r^2$. При больших значениях r эта простая оценка хорошо аппроксимирует величину $L(r+1, 1-\alpha)$.

2.4.2. Метод переписи Шнабеля

Один из методов, используемых в экологии для оценки численности популяций, носит название *переписи Шнабеля* [264, 280]. Из популяции с неизменным составом производится отлов заранее заданного числа особей, каждая из которых имеет константную вероятность отлова. При отлове особь помечается (если она не была помечена ранее) и возвращается обратно в популяцию. Статистические оценки численности популяции вычисляются на основе данных о числе помеченных особей среди отловленных.

Для оценки числа локальных оптимумов метод переписи Шнабеля адаптируется следующим образом. Выполняются r испытаний локального поиска из начальных точек, выбранных случайным образом с одинаковым распределением (число r выбирается априори). Пусть K – случайная величина, равная числу различных локальных оптимумов, полученных за r испытаний. Обозначим через $\mathbf{F}_{\mathbf{p}}(k; \nu, r) = \mathbf{P}\{K \leq k\}$ функцию распределения величины K в предположении о том, что $\mathbf{p} = (p_1, \ldots, p_\nu)$ – вектор вероятностей попадания в локальные оптимумы.

Для вычисления доверительных интервалов на параметр ν известны приближенные методы, в которых используется аппроксимация распреде-

ления величины K нормальным распределением (см., например, [264,270]), а также точные методы [191,258], основанные на непосредственном вычислении функции распределения, подобно выражениям из утверждения 2.10.

Как было замечено в [150], в изотропном случае распределение вероятностей случайной величины K имеет вид:

$$\mathbf{P}\{K=k\} = \frac{\nu(\nu-1)...(\nu-k+1)\ S(r,k)}{\nu^r} = \frac{\nu!\ S(r,k)}{(\nu-k)!\ \nu^r},$$

где $S(r,k)$ – число Стирлинга второго рода, равное количеству способов разбить r-элементное множество на k непустых подмножеств без учета порядка подмножеств, $S(r,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k+j}\binom{k}{j}j^r$.

Найденные в [154] асимптотические оценки дисперсии позволяют построить консервативный доверительный интервал, используя упрощающее предположение о нормальности распределения данной оценки. Как показано в [191, 258], более точно вычислить консервативный доверительный интервал удается при использовании нормальной аппроксимации распределения K. Однако оба подхода могут давать существенные ошибки в ситуациях, когда предположение нормальности оказывается неуместным.

Рассмотрим следствие основного результата А.М. Зубкова и Н.Н. Попова из [54], на основе которого будет получен аналог утверждения 2.12 для переписи Шнабеля. Пусть имеется r частиц, размещаемых случайным образом, независимо, с одинаковым распределением по ячейкам $1,2,\ldots,\nu$. Вероятность попадания в j-ю ячейку равна p_j, $j=1,\ldots,\nu$, $\sum_{j=1}^{\nu}p_j=1$. Как показано в [54], в случае изотропного распределения вероятностей $(p_1,\ldots,p_\nu)=\overline{\mathbf{p}}$ число занятых ячеек $\mu(r,\overline{\mathbf{p}})$ «стохастически больше» числа занятых ячеек $\mu(r,\mathbf{p}')$ при любом другом распределении вероятностей $\mathbf{p}'$, т. е. $\mathbf{P}\{\mu(r,\overline{\mathbf{p}})\le k\} \le \mathbf{P}\{\mu(r,\mathbf{p}')\le k\}$ для всех $r\in\mathbb{N}$, $k=1,\ldots,\nu$.

Обозначая через $\mathbf{F}_{\mathbf{p}}(k;r,\nu)$ функцию распределения случайной величины K при вероятностях попадания в локальные оптимумы $p_1,\ldots,p_\nu$, $\mathbf{p}=(p_1,\ldots,p_\nu)$, указанный выше результат из [54] записывается как

Утверждение 2.13. *При любых фиксированных* $\nu \in \mathbb{N}$, $r \in \mathbb{N}$, $k = 0, \ldots, r$, *функция* $\mathbf{F}_{\mathbf{p}}(k; r, \nu)$ *достигает минимума в точке* $\mathbf{p} = \overline{\mathbf{p}}$.

Утверждение 2.13 позволяет вычислять консервативную нижнюю границу $L'(r, k, \mathbf{p})$ для ν с заданным доверительным уровнем p ввиду утверждения 2.11. Корректность такой нижней границы при условии справедливости утверждения 2.13 была отмечена в [181]. Аналогичная нижняя граница предложена в работе [238] Ч. Мао. Заметим, что $\mathbf{F}_{\mathbf{p}}(r; \nu, r) = 1$ при любом ν, поэтому при $k = r$ нижняя оценка $L'(r, k, \mathbf{p})$ не определена. Подобно доказательству следствия 2.4, из утверждения 2.13 получаем

Следствие 2.5. $\mathbf{F}_{\overline{\mathbf{p}}}(k; r, \nu)$ *является невозрастающей функцией от* ν *при любых* $r \in \mathbb{N}$, $k = 0, \ldots, r$.

Последнее следствие было независимо доказано в [181, 258] без использования результата [54]. Следствие 2.5 дает основание применению метода дихотомии для построения нижней границы $L'(r, k, \mathbf{p})$ в соответствии с утверждением 2.11, а в изотропном случае – консервативного доверительного интервала в соответствии с утверждением 2.10.

Заключение

В [181] предложенные методы оценивания числа локальных оптимумов были опробованы на ряде задач комбинаторной оптимизации. Среди них задачи с целевыми функциями, построенными случайным образом по модели С. Кауффмана [225], задачи отыскания двоичной последовательности с минимальной автокорреляцией [198], задачи о вершинном покрытии и др. Проведенные вычислительные эксперименты показали адекватность предложенных нижних оценок для числа локальных оптимумов. В то же время двусторонние доверительные интервалы зачастую не покрывают истинное значение оцениваемого параметра, что говорит о неприменимости

к рассматриваемым задачам предположения о равновероятном попадании в локальные оптимумы алгоритма локального поиска.

Для оценивания числа локальных оптимумов снизу в большинстве случаев предпочтительнее использовать нижнюю границу, основанную на методе переписи Шнабеля, так как нижняя граница, основанная на времени первого повторения локальных оптимумов, дает нестабильные результаты. Тем не менее последняя оценка может оказаться единственным приемлемым методом, когда локальные оптимумы повторяются чрезвычайно редко и априори сложно выбрать число испытаний, при котором будет получено хотя бы одно повторение ранее полученного локального оптимума.

Дальнейшие исследования могут быть направлены на использование предложенных методов в правилах остановки для эвристик локального поиска с многократным перезапуском. В частности, необходимое условие для остановки алгоритма может формулироваться как условие превышения числа найденных локальных оптимумов над нижней границей заданного доверительного уровня. Также предложенные методы могут использоваться для классификации тестовых примеров по их сложности для алгоритмов, основанных на локальном поиске. Новые приложения методов оценивания числа локальных оптимумов наметились в биоинформатике при исследовании свойств молекул РНК [263].

Для некоторых классических задач комбинаторной оптимизации известны специальные точные и приближенные методы перечисления локальных оптимумов, например, неприводимых покрытий [30] и максимальных паросочетаний [223]. Представляет интерес проведение экспериментального сравнения этих методов с предложенными в данном разделе.

Доказательство утверждения 2.12 получено автором совместно с С.А. Клоковым. Идея использования метода переписи Шнабеля и ожидания первого повторения для оценивания числа локальных оптимумов принадлежит К.Р. Ривсу.

3. Исследование сложности задачи оптимальной рекомбинации

3.1. Постановка задачи

Работоспособность генетического алгоритма существенно зависит от выбора оператора кроссинговера, где комбинируются элементы родительских решений при построении особей-потомков [220]. В данной главе исследуются сложность и способы решения *задачи оптимальной рекомбинации* (ЗОР), состоящей в отыскании наилучшего возможного результата кроссинговера при заданных двух родительских генотипах, представляющих допустимые решения задачи комбинаторной оптимизации. ЗОР является вспомогательной задачей, как правило, меньшей размерности, чем исходная задача, и формулируется с учетом основных принципов построения оператора кроссинговера [261]. Результаты, содержащиеся в [42, 108, 122, 160, 195] и других работах, дают экспериментальное подтверждение целесообразности решения (точного или приближенного) ЗОР в операторах кроссинговера генетических алгоритмов.

Особый интерес представляют NP-трудные задачи, для которых ЗОР полиномиально разрешима. Впервые такие задачи были обнаружены в [108] и [122]. В этих работах эффективные операторы кроссинговера, осуществляющие оптимальную рекомбинацию, были предложены и реализованы в ГА для задач о наибольшем независимом множестве графа и наибольшей клике.

В настоящей главе ЗОР рассматривается в предположении двоичного представления решений, совпадающего с кодировкой решений задачи комбинаторной оптимизации, поэтому термин «генотип» будет обозначать

элемент множества допустимых решений $\mathrm{Sol} \subseteq \{0,1\}^{n(I)}$.

Вообще говоря, индивидуальная задача из задачи комбинаторной оптимизации может не иметь допустимых решений. Однако такие частные случаи не представляют интереса с точки зрения задачи оптимальной рекомбинации, так как в этой задаче предполагается наличие двух (возможно, совпадающих между собой) допустимых родительских решений, являющихся частью входных данных задачи. В связи с этим далее будем предполагать, что для рассматриваемых индивидуальных задач множество допустимых решений не пусто.

Определение 3.1. *Задачей оптимальной рекомбинации для задачи комбинаторной оптимизации* $\Pi = (\mathrm{Inst}, \mathrm{Sol}, f)$ *является задача комбинаторной оптимизации* $\overline{\Pi} = (\overline{\mathrm{Inst}}, \overline{\mathrm{Sol}}, \overline{f})$, *такая что всякая индивидуальная задача* $\overline{I} \in \overline{\mathrm{Inst}}$ *имеет вид* $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2)$, *где* $I \in \mathrm{Inst}$, $\mathbf{p}^1 = (p^1_1, \ldots, p^1_{n(I)}) \in \mathrm{Sol}(I)$, $\mathbf{p}^2 = (p^2_1, \ldots, p^2_{n(I)}) \in \mathrm{Sol}(I)$, *и*

$$\overline{\mathrm{Sol}}(\overline{I}) = \{\mathbf{x} \in \mathrm{Sol}(I) |\ x_j = p^1_j \text{ или } x_j = p^2_j,\ j = 1, \ldots, n(I)\}. \qquad (3.1)$$

Критерий оптимизации в $\overline{I}$ *тот же, что и в* I, *т. е.* $\overline{f}_{\overline{I}} \equiv f_I$.

Допустимые решения $\mathbf{p}^1, \mathbf{p}^2$ задачи I называются *родительскими* решениями для задачи $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2)$. Далее множество номеров координат, в которых родительские решения различны, будем обозначать через $D(\mathbf{p}^1, \mathbf{p}^2)$, т. е. $D(\mathbf{p}^1, \mathbf{p}^2) = \{j : p^1_j \neq p^2_j\}$.

Определение 3.1 обеспечивает выполнение свойства передачи генов [261] (см. п. 1.2.1) и может интерпретироваться как «фиксация» в ЗОР тех значений генов, в которых оба родительских генотипа совпадают. В литературе, однако, известны и другие постановки задач рекомбинации, где в соответствии с терминологией из [148] осуществляется *аллельная династически оптимальная рекомбинация*. Например, в работах [15,34,134,147,237] хорошие экспериментальные результаты показали ГА, в которых решается

задача рекомбинации с «фиксацией» только тех значений генов, в которых оба родительских генотипа имеют значение, равное 0. Такая постановка задачи далее будет называться *ослабленной задачей оптимальной рекомбинации* и для ее формулировки достаточно заменить условие (3.1) в определении 3.1 на

$$\overline{\mathrm{Sol}}(\overline{I}) = \{\mathbf{x} \in \mathrm{Sol}(I) |\ x_j \leq p_j^1 \text{ или } x_j \leq p_j^2,\ j = 1, \ldots, n(I)\}. \qquad (3.2)$$

Применение ослабленной задачи оптимальной рекомбинации в операторе кроссинговера рассматривается в главе 5.

В ряде ГА при решении NP-трудных задач рекомбинации используются методы ветвей и границ [134,160] или динамического программирования [294]. В операторе кроссинговера зачастую применяются приближенные варианты этих точных методов во избежание чрезмерно высоких вычислительных затрат на этапе рекомбинации. Так, в случае использования динамического программирования мощность множества состояний может ограничиваться некоторой заданной заранее величиной [294], а в случае использования метода ветвей и границ результатом рекомбинации может быть рекордное решение, полученное этим методом за ограниченное время или ограниченное число итераций [160].

Также может применяться метод снижения размерности задачи рекомбинации за счет группировки переменных в «блоки», когда значения целого блока переменных переходят в решение-потомок либо от одного родительского решения, либо от другого [149]. Помимо сокращенных вариантов точных методов для приближенного решения задачи оптимальной рекомбинации могут использоваться известные приближенные эвристики. Например, в генетическом алгоритме для задачи вершинного покрытия [15] при решении ослабленной ЗОР используется жадный алгоритм П. Эрдеша, а в известном алгоритме *связывающего пути* [195] предлагается выполнять частичный перебор допустимых решений ЗОР вдоль некоторого пути, соединяющего между собой пару родительских решений.

В настоящей главе ряд задач комбинаторной оптимизации рассматриваются как задачи булевого линейного программирования:

$$\max \quad f(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j, \tag{3.3}$$

$$\sum_{j=1}^{n} q_{ij} x_j \leq b_i, \quad i = 1, \ldots, m, \tag{3.4}$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n. \tag{3.5}$$

Здесь $\mathbf{x} = (x_1, \ldots, x_n) \in \{0, 1\}^n$ – вектор булевых переменных, а исходные данные задачи c_j, $j = 1, \ldots, n$; q_{ij}, $i = 1, \ldots, m$; $j = 1, \ldots, n$, и b_i, $i = 1, \ldots, m$, принадлежат множеству рациональных чисел $\mathbb{Q}$. Задачи, где вместо знака «$\leq$» в условии (3.4) имеет место «$\geq$» или «=» для некоторых значений i (или для всех $i = 1, \ldots, m$), могут быть легко преобразованы в задачу вида (3.3)–(3.5). При этом число ограничений увеличится не более чем в два раза. Задачи минимизации могут быть представлены аналогичным образом при использовании коэффициентов c_j с обратным знаком. Если уместно, будем пользоваться более компактной формой записи для задачи (3.3)–(3.5):

$$\max \left\{ \mathbf{cx} : \mathbf{Qx} \leq \mathbf{b}, \mathbf{x} \in \{0, 1\}^n \right\},$$

где $(m \times n)$-матрица $\mathbf{Q}$ имеет элементы q_{ij}, $i = 1, \ldots, m, j = 1, \ldots, n$; $\mathbf{b} = (b_1, \ldots, b_m)$; $\mathbf{c} = (c_1, \ldots, c_n)$.

Глава имеет следующую структуру. С использованием эффективных сводимостей между задачами оптимальной рекомбинации в разделе 3.2 доказывается, что оптимальная рекомбинация полиномиально вычислима для задач упаковки множества наибольшего веса, разбиения множества минимального веса и для одного варианта простейшей задачи размещения производства. Здесь также предлагается полиномиально вычислимый оператор оптимальной рекомбинации для задач булевого линейного программирования, когда в каждое ограничение входят не более двух переменных, что обобщает результаты из [108] и [122].

В разделе 3.3 установлена NP-трудность ряда задач оптимальной рекомбинации. Во-первых, доказана NP-трудность оптимальной рекомбинации для задач булевого линейного программирования с тремя переменными в каждом ограничении. Затем показана NP-трудность ЗОР для задачи об одномерном булевом рюкзаке, задач о покрытии и p-медиане, а также для задачи максимальной выполнимости. Доказана NP-трудность ЗОР для задачи коммивояжера в симметрическом и в общем случаях, рассмотрены подходы к их решению.

3.2. Полиномиально разрешимые случаи

3.2.1. Задачи о независимом множестве, вершинном покрытии и клике

Пусть дан граф $G = (V, E)$ с множеством вершин $V = \{v_1, \ldots, v_{|V|}\}$ и множеством ребер E и указаны положительные веса вершин $c_v, \ \ v \in V$.

Задача о независимом множестве состоит в отыскании подмножества вершин $S \subseteq V$, такого что никакое ребро $e \in E$ не инцидентно сразу двум вершинам из S (т. е. S – независимое множество в графе G) и вес $\sum_{v \in S} c_v$ множества S максимален.

Задача о независимом множестве NP-трудна [27] и эквивалентна задаче о вершинном покрытии (см. раздел 1.1). Также рассматриваемая задача эквивалентна известной *задаче о клике*, которая состоит в отыскании подмножества вершин $Q \subseteq V$, такого что любые две вершины u, v из Q смежны и вес множества Q максимален.

Будем предполагать, что кодировка решений из множества Sol такова, что $n = |V|$, и при любом $j, \ j = 1, \ldots, n$, имеем $x_j = 1$, если вершина v_j принадлежит кодируемому подмножеству, иначе $x_j = 0$. Следующая теорема сводит воедино результаты, полученные в [108], [121] и [122].

Теорема 3.1. [108, 121, 122] *ЗОР для задач о независимом множестве,*

клике и вершинном покрытии разрешимы за время $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n)$.

Доказательство. Приведем доказательство только для задачи о независимом множестве, так как рассуждения для двух других задач аналогичны.

Рассмотрим произвольные родительские независимые множества S_1 и S_2 и соответствующие им генотипы $\mathbf{p}_1$ и $\mathbf{p}_2$. Решение-потомок S должно содержать все множество вершин $L = S_1 \cap S_2$, кроме того, в S не должно быть элементов множества $V \setminus (S_1 \cup S_2)$, а вершины с номерами из множества $D(\mathbf{p}^1, \mathbf{p}^2)$ необходимо выбрать оптимальным образом. Последнее требование формулируется как задача о наибольшем независимом множестве в подграфе $H = (V', E')$, порожденном множеством вершин с номерами из $D(\mathbf{p}^1, \mathbf{p}^2)$. Легко видеть, что данный подграф является двудольным. Пусть $V' = V'_1 \cup V'_2$, где V'_1, V'_2 – множества вершин, образующих доли графа H. Для отыскания независимого множества наибольшего веса в H достаточно найти вершинное покрытие C' минимального веса в H и перейти к его дополнению.

Задача о наименьшем вершинном покрытии двудольного графа H эффективно разрешима с помощью алгоритма построения минимального разреза (см., например, [212]). Для этого рассмотрим вспомогательную сеть $\mathcal{N}$, множество вершин которой состоит из V' и дополнительных вершины-источника v_0 и вершины-стока v_{n+1}. Пусть в сети $\mathcal{N}$ из вершины v_0 выходят дуги во все вершины доли V'_1, кроме того, пусть в вершину-сток v_{n+1} идут дуги от вершин доли V'_2. Пропускные способности всех этих дуг полагаются равными весам инцидентных с ними вершин графа H. Каждое ребро uv из множества E' определяет в $\mathcal{N}$ дугу между вершинами u и v, направленную из V'_1 в V'_2. Пропускная способность такой дуги полагается равной $\max\{c_u, c_v\}$.

Построенная задача о минимальном разрезе с источником v_0 и стоком v_{n+1} разрешима за время $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3)$ с помощью алгоритма

А.В. Карзанова – см., например, [86]. Не теряя общности, можно считать, что разрез проходит только по дугам, выходящим из v_0 или входящим в v_{n+1}, так как если некоторая дуга (u, v), $u \in V_1', v \in V_2'$, войдет в разрез, то эта дуга может быть заменена одной из дуг (v_0, u) или (v, v_{n+1}) без увеличения веса разреза.

Вершинное покрытие наименьшего веса C' формируется из вершин, инцидентных дугам минимального разреза. Генотип ξ, являющийся вектором-индикатором множества вершин $L \cup (V' \setminus C')$, представляет собой решение задачи оптимальной рекомбинации. Общая тудоемкость решения ЗОР составляет $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n)$. □

3.2.2. Сводимости задач оптимальной рекомбинации

Типичный способ распространения результатов об эффективной разрешимости или труднорешаемости задач в теории сложности основывается на построении эффективно вычислимых сводимостей от задач с известным статусом сложности к новым задачам. Для применения данного подхода к задачам оптимальной рекомбинации сформулируем сначала достаточно общий признак сводимости задач комбинаторной оптимизации.

Утверждение 3.1. *Пусть* $\Pi_1 = (\mathrm{Inst}_1, \mathrm{Sol}_1, f_I)$, $\Pi_2 = (\mathrm{Inst}_2, \mathrm{Sol}_2, g_{I'})$ *– задачи комбинаторной оптимизации на максимум (на минимум) и существуют отображение* $\alpha : \mathrm{Inst}_1 \to \mathrm{Inst}_2$ *и инъекция* $\beta : \mathrm{Sol}_1(I) \to \mathrm{Sol}_2(\alpha(I))$, *такие что при* $I \in \mathrm{Inst}_1$*:*

1) для любых $\mathbf{x}, \mathbf{x}' \in \mathrm{Sol}_1(I)$, *удовлетворяющих условию*

$$f_I(\mathbf{x}) > f_I(\mathbf{x}'), \tag{3.6}$$

имеет место неравенство

$$g_{\alpha(I)}(\beta(\mathbf{x})) > g_{\alpha(I)}(\beta(\mathbf{x}')) \tag{3.7}$$

(если Π_1 *– задача минимизации, то в (3.6) знак неравенства меняется на "*$<$*"; если* Π_2 *– задача минимизации, то в (3.7) знак неравенства меняется на "*$<$*");*

2) *если* $\mathbf{y} \in \beta(\mathrm{Sol}_1(I))$, $\mathbf{y}' \in \mathrm{Sol}_2(\alpha(I))$ *и*

$$g_{\alpha(I)}(\mathbf{y}') \geq g_{\alpha(I)}(\mathbf{y}), \tag{3.8}$$

то $\mathbf{y}' \in \beta(\mathrm{Sol}_1(I))$ *(если* Π_2 *– задача минимизации, то знак неравенства в формуле (3.8) меняется на "*$\leq$*").*

Тогда Π_1 *сводится к* Π_2*, причем любая индивидуальная задача* $I \in \mathrm{Inst}_1$ *разрешима за время* $O(T_\alpha(I) + T_{\beta^{-1}}(I) + T(I))$*, где* $T_\alpha(I)$ *– трудоемкость вычисления* $\alpha(I)$*;* $T_{\beta^{-1}}(I)$ *– оценка сверху трудоемкости вычисления* $\beta^{-1}(\mathbf{y})$, $\mathbf{y} \in \beta(\mathrm{Sol}(I))$*;* $T(I)$ *– время решения индивидуальной задачи* $\alpha(I)$*.*

Доказательство. Пусть $I \in \mathrm{Inst}_1$. Рассмотрим оптимум $\mathbf{y}^*$ индивидуальной задачи $\alpha(I)$. Согласно условию 2, при $\mathrm{Sol}_1(I) \neq \emptyset$, решение $\mathbf{y}^*$ принадлежит множеству $\beta(\mathrm{Sol}_1(I))$. Рассуждая от противного, с учетом условия 1, заключаем что если $\mathrm{Sol}_1(I) \neq \emptyset$, то $\beta^{-1}(\mathbf{y}^*)$ – оптимум для I. □

Если трудоемкости $T_\alpha(I), T_{\beta^{-1}}(I)$ и $T(I)$ полиномиально ограничены относительно $|I|$, то утверждение 3.1 дает признак полиномиальной сводимости одной задачи класса NPO к другой. Подобный признак полиномиальной сводимости использован, например, в работе А.А. Агеева [1].

Следующее утверждение имеет целью построение эффективных сводимостей от одной ЗОР к другой, когда имеются эффективные сводимости соответствующих задач комбинаторной оптимизации.

Утверждение 3.2. *Пусть* $\Pi_1 = (\mathrm{Inst}_1, \mathrm{Sol}_1, f_I)$ *и* $\Pi_2 = (\mathrm{Inst}_2, \mathrm{Sol}_2, g_{I'})$ *являются задачами комбинаторной оптимизации, где* $\mathrm{Sol}_1(I) \subseteq \{0,1\}^{n_1(I)}$, $\mathrm{Sol}_2(I') \subseteq \{0,1\}^{n_2(I')}$*, и существуют отображения* α *и* β*, для которых выполняется признак сводимости* Π_1 *к* Π_2 *из утверждения 3.1, причем*

(i) для любого $j = 1, \ldots, n_1(I)$ *существует такое* $k(j)$, *что* $\beta^{-1}(\mathbf{y})_j$ *– функция от* $y_{k(j)}$ *при* $\mathbf{y} = (y_1, \ldots, y_{n_2}) \in \beta(\mathrm{Sol}_1(I))$;

(ii) для любого $k = 1, \ldots, n_2(\alpha(I))$ *существует такое* $j(k)$, *что* $\beta(\mathbf{x})_k$ *– функция от* $x_{j(k)}$ *при* $\mathbf{x} = (x_1, \ldots, x_{n_1}) \in \mathrm{Sol}_1(I)$.

Тогда $\overline{\Pi}_1$ *сводится к* $\overline{\Pi}_2$, *причем любая индивидуальная задача* $\overline{I} = (I, \mathbf{p}^1, \mathbf{p}^2)$ *из ЗОР* $\overline{\Pi}_1$ *разрешима за время* $O(T_\alpha(I) + T_\beta(I) + T_{\beta^{-1}}(I) + \overline{T}(I, \mathbf{p}^1, \mathbf{p}^2))$, *где* $\overline{T}(I, \mathbf{p}^1, \mathbf{p}^2)$ *– время решения ЗОР* $(\alpha(I), \beta(\mathbf{p}^1), \beta(\mathbf{p}^2))$, *а* $T_\beta(I)$ *– оценка сверху времени вычисления* $\beta(\mathbf{x})$, $\mathbf{x} \in \mathrm{Sol}(I)$.

Доказательство. Не теряя общности, будем предполагать, что Π_1 и Π_1 – задачи максимизации. Пусть имеется индивидуальная задача I из задачи Π_1 и два ее допустимых решения $\mathbf{p}^1, \mathbf{p}^2$. Этим решениям соответствуют допустимые решения $\mathbf{q}^1 = \beta(\mathbf{p}^1)$, $\mathbf{q}^2 = \beta(\mathbf{p}^2)$ задачи $\alpha(I)$.

Рассмотрим ЗОР для индивидуальной задачи $\alpha(I) \in \Pi_2$ с родительскими решениями $\mathbf{q}^1, \mathbf{q}^2$. Оптимальное решение этой ЗОР $\mathbf{y} \in \mathrm{Sol}_2(\alpha(I))$ может быть преобразовано за время $T_{\beta^{-1}}$ в допустимое решение $\mathbf{z} = \beta^{-1}(\mathbf{y}) \in \mathrm{Sol}_1(I)$ задачи Π_1.

Заметим, что для всех $j \notin D(\mathbf{p}^1, \mathbf{p}^2)$ имеют место равенства $z_j = p^1_j = p^2_j$. Действительно, по условию (i) для любого $j = 1, \ldots, n_1(I)$ найдется такое $k(j)$, что

(I) либо $\beta^{-1}(\mathbf{y})_j = y_{k(j)}$ для всех $\mathbf{y} \in \beta(\mathrm{Sol}_1(I))$, либо

(II) $\beta^{-1}(\mathbf{y})_j = 1 - y_{k(j)}$ для всех $\mathbf{y} \in \beta(\mathrm{Sol}_1(I))$, либо

(III) $\beta^{-1}(\mathbf{y})_j$ постоянна на множестве $\beta(\mathrm{Sol}_1(I))$.

В случае (I) для всех $j \notin D(\mathbf{p}^1, \mathbf{p}^2)$ имеем $z_j = y_{k(j)}$. Кроме того, $y_{k(j)} = q^1_{k(j)} = q^2_{k(j)}$ по определению ЗОР, ибо $q^1_{k(j)} = p^1_j = p^2_j = q^2_{k(j)}$. Следовательно, $z_j = q^1_{k(j)} = p^1_j = p^2_j$.

Случай (II) рассматривается аналогично.

Случай (III) тривиален, так как $\mathbf{z}, \mathbf{p}^1, \mathbf{p}^2 \in \beta^{-1}(\beta(\mathrm{Sol}_1(I)))$, т. е. координата z_j неизменна и совпадает с p^1_j и p^2_j.

Для доказательства оптимальности $\mathbf{z}$ в индивидуальной задаче $\overline{I}$ из

ЗОР $\overline{\Pi}_1$ предположим от противного, что существует такое допустимое решение $\mathbf{z}' = (z'_1, \ldots, z'_{n_1}) \in \mathrm{Sol}_1(I)$, что $z'_j = p^1_j = p^2_j$ для всех $j \notin D(\mathbf{p}^1, \mathbf{p}^2)$ и $f_I(\mathbf{z}') > f_I(\mathbf{z})$. Тогда $g_{\alpha(I)}(\beta(\mathbf{z}')) > g_{\alpha(I)}(\beta(\mathbf{z})) = g_{\alpha(I)}(\mathbf{y})$. Однако, как следует из условия (ii), $\beta(\mathbf{z}')$ совпадает с $\mathbf{q}^1$ во всех координатах $k \notin D(\mathbf{q}^1, \mathbf{q}^2)$ (чтобы в этом убедиться достаточно рассмотреть три случая аналогичных (I) – (III)). Таким образом, $\mathbf{y}$ не является оптимальным решением индивидуальной задачи оптимальной рекомбинации для $\alpha(I)$, что приводит к противоречию. $\square$

Наиболее применимым утверждение 3.2 представляется в частном случае, когда ЗОР $\overline{\Pi}_2$ полиномиально разрешима, преобразования α, β и β^{-1} эффективно вычислимы, $n_1(I) \equiv n_2(I')$ и $k(j) \equiv j,\ j(k) \equiv k$.

3.2.3. Задачи упаковки, разбиения и размещения как задачи булевого линейного программирования

Воспользуемся утверждением 3.2 с целью получения эффективного оператора оптимальной рекомбинации для задачи упаковки множества, которая в постановке булевого линейного программирования имеет вид:

$$\max\left\{f_{\mathrm{pack}}(\mathbf{x}) = \mathbf{cx} : \mathbf{Qx} \leq \mathbf{e}, \mathbf{x} \in \{0,1\}^n\right\}, \tag{3.9}$$

где $\mathbf{Q}$ – заданная матрица размерности $m \times n$ из нулей и единиц; $\mathbf{e}$ – вектор из m единиц. Известная сводимость задачи упаковки множества к задаче о независимом множестве основана на следующем преобразовании α. Построим граф на множестве вершин $v_1, \ldots, v_n$ с весами $c_1, \ldots, c_n$. Ребрами соединим пары вершин v_j, v_k в тех и только тех случаях, когда j и k вместе принадлежат по крайней мере одному из подмножеств $U_i = \{j : q_{ij} = 1, j = 1, \ldots, n\}, i = 1, \ldots, m$. В данном случае β представляет собой тождественное отображение, а целевые функции f и g совпадают. Из утверждения 3.2 получаем

Следствие 3.1. *ЗОР для задачи упаковки множества* (3.9) *разрешима за время* $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n^2 m)$.

Во многих сводимостях задач комбинаторной оптимизации множеству допустимых решений исходной индивидуальной задачи I сопоставляется подмножество допустимых решений $\beta(\mathrm{Sol}_1(I)) \subseteq \mathrm{Sol}_2(\alpha(I))$ со значениями целевой функции, не уступающими некоторому пороговому значению. Сводимости такого вида использованы в доказательстве полиномиальной разрешимости ЗОР для двух следующих задач.

Задача разбиения множества: найти

$$\min \{ f_{\mathrm{part}}(\mathbf{x}) = \mathbf{cx} : \mathbf{Qx} = \mathbf{e},\ \mathbf{x} \in \{0,1\}^n \}, \tag{3.10}$$

где $\mathbf{Q}$ – заданная $(m \times n)$-матрица из нулей и единиц.

Простейшая задача размещения производства в виде задачи булевого линейного программирования: найти

$$\min \quad f_{\mathrm{splp}}(\mathbf{Y}, \mathbf{u}) = \sum_{k=1}^{K} \sum_{\ell=1}^{L} c_{k\ell} y_{k\ell} + \sum_{k=1}^{K} C_k u_k, \tag{3.11}$$

$$\sum_{k=1}^{K} y_{k\ell} = 1, \quad \ell = 1, \ldots, L, \tag{3.12}$$

$$u_k \geq y_{k\ell}, \quad k = 1, \ldots, K,\ \ell = 1, \ldots, L, \tag{3.13}$$

$$y_{k\ell} \in \{0,1\},\ u_k \in \{0,1\}, \quad k = 1, \ldots, K,\ \ell = 1, \ldots, L. \tag{3.14}$$

Здесь и далее $(K \times L)$-матрица из булевых переменных $y_{k\ell}$ обозначается через $\mathbf{Y}$, а K-мерный вектор из булевых переменных u_k – через $\mathbf{u}$. Стоимости $c_{k\ell} \in \mathbb{Q}$, $C_k \in \mathbb{Q}$ предполагаются неотрицательными.

Простейшая задача размещения производства в виде задачи булевого линейного программирования полиномиально эквивалентна задаче (1.1) – (1.2) и, более того, содержательно представляет собой ту же самую задачу

(см., например, [10]). Однако согласно с терминологией из определения 1.1 задача (3.11)–(3.14) с представлением решений в виде пары $(\mathbf{Y}, \mathbf{u})$ соответствует задаче комбинаторной оптимизации, отличной от задачи (1.1)–(1.2). Как будет показано в разделе 3.3, ЗОР для последней задачи NP-трудна.

Докажем, что ЗОР для простейшей задачи размещения производства в виде задачи булевого линейного программирования (3.11)–(3.14) и для задачи разбиения множества (3.10) эффективно разрешимы.

Следствие 3.2. *(i) ЗОР для задачи разбиения множества (3.10) разрешима за время* $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + n^2 m)$.

(ii) ЗОР для простейшей задачи размещения производства в виде задачи булевого линейного программирования (3.11)–(3.14) разрешима за полиномиальное время.

Доказательство. В обоих пунктах доказательства будут использованы известные сводимости соответствующих задач комбинаторной оптимизации к задаче упаковки множества (см., например, сводимости T2 и T5 в [232]).

(i) Пусть задача разбиения множества обозначается через Π_1, а Π_2 – задача упаковки множества. Вход ЗОР для Π_1 состоит из индивидуальной задачи $I \in \mathrm{Inst}_1$ и двух родительских решений, поэтому $\mathrm{Sol}_1(I) \neq \emptyset$ и задача I эквивалентна следующей задаче частично целочисленного линейного программирования

$$\min \quad \sum_{j=1}^{n} c_j x_j + \lambda \sum_{i=1}^{m} w_i, \tag{3.15}$$

$$\sum_{j=1}^{n} q_{ij} x_j + w_i = 1, \ i = 1, \ldots, m, \tag{3.16}$$

$$x_j \in \{0, 1\}, \ j = 1, \ldots, n; \quad w_i \geq 0, \ i = 1, \ldots, m, \tag{3.17}$$

где $\lambda > 2 \sum_{j=1}^{n} |c_j|$ – штрафной коэффициент, достаточный, чтобы обеспечить обращение в ноль всех вспомогательных переменных w_i в оптималь-

ном решении. Подстановка переменных w_i в целевую функцию преобразует (3.15)–(3.17) в задачу

$$\min\left\{\lambda m + \sum_{j=1}^{n}\left(c_j - \lambda\sum_{i=1}^{m} q_{ij}\right)x_j \mid \mathbf{Qx} \le \mathbf{e},\ \mathbf{x}\in\{0,1\}^n\right\},$$

которая эквивалентна следующей индивидуальной задаче $\alpha(I)$ из задачи упаковки множества:

$$\max\left\{g(\mathbf{x}) = \sum_{j=1}^{n}\left(\lambda\sum_{i=1}^{m} q_{ij} - c_j\right)x_j \mid \mathbf{Qx} \le \mathbf{e},\ \mathbf{x}\in\{0,1\}^n\right\}.$$

Пусть β – тождественное отображение. Тогда каждое допустимое решение $\mathbf{x}$ задачи разбиения множества является допустимым решением для задачи Π_2 со значением целевой функции $g(\mathbf{x}) = \lambda m - f_{\text{part}}(\mathbf{x}) > \lambda(m-1/2)$. В то же время, если вектор $\mathbf{x}'$ является допустимым решением задачи Π_2, но не является допустимым для задачи разбиения множества, то его значение целевой функции есть $g(\mathbf{x}') = \lambda(m-k) - f_{\text{part}}(\mathbf{x}')$, где k – число ограничений вида $\sum_{j=1}^{n} q_{ij}x_j = 1$, нарушенных вектором $\mathbf{x}'$. Таким образом, β является биекцией из $\mathrm{Sol}_1(I)$ на множество допустимых решений с достаточно большим значением целевой функции:

$$\{\mathbf{x}\in\mathrm{Sol}_2(\alpha(I)) \mid g(\mathbf{x}) > \lambda(m-1/2)\}.$$

Оценку трудоемкости решения ЗОР для Π_2 дает следствие 3.1. Применение утверждения 3.2 завершает доказательство пункта (i).

(ii) Пусть Π_1' – простейшая задача размещения производства. Аналогично доказательству пункта (i) преобразуем уравнения (3.12) в неравенства. С этой целью запишем (3.12) в виде $\sum_{k=1}^{K} y_{k\ell} + w_\ell = 1,\ \ell = 1,\ldots,L$, с использованием неотрицательных вспомогательных переменных w_ℓ и обеспечим их равенство нулю в оптимальном решении, прибавив к критерию (3.11) функцию штрафа $\lambda\sum_{\ell=1}^{L} w_\ell$. Здесь

$$\lambda > \sum_{k=1}^{K} C_k + \sum_{\ell=1}^{L}\max_{k=1,\ldots,K} c_{k\ell}.$$

Исключая переменные w_ℓ, заменяем условие (3.12) на $\sum_{k=1}^{K} y_{k\ell} \le 1,\ \ell = 1,\dots,L$, а функцию штрафа – на $\lambda L - \lambda \sum_{\ell=1}^{L}\sum_{k=1}^{K} y_{k\ell}$. Умножая на -1 целевую функцию и переходя к новым переменным $\overline{u}_k = 1 - u_k,\ k = 1,\dots,K$, получаем следующую задачу максимизации Π_2':

$$\max\ \ g'(\mathbf{Y},\overline{\mathbf{u}}) = \sum_{k=1}^{K}\sum_{\ell=1}^{L}(\lambda - c_{k\ell})y_{k\ell} + \sum_{k=1}^{K} C_k\overline{u}_k - \lambda L - \sum_{k=1}^{K} C_k, \qquad (3.18)$$

$$\sum_{k=1}^{K} y_{k\ell} \le 1, \quad \ell = 1,\dots,L, \qquad (3.19)$$

$$\overline{u}_k + y_{k\ell} \le 1, \quad k = 1,\dots,K,\ \ell = 1,\dots,L, \qquad (3.20)$$

$$y_{k\ell} \in \{0,1\},\ \overline{u}_k \in \{0,1\}, \quad k = 1,\dots,K,\ \ell = 1,\dots,L, \qquad (3.21)$$

где $\overline{\mathbf{u}} = (\overline{u}_1,\dots,\overline{u}_K)$. Очевидно, Π_2' представляет собой частный случай задачи упаковки множества, если из целевой функции вычесть константу $-\lambda L - \sum_{k=1}^{K} C_k$. Таким образом, отображение $\alpha(I)$ полностью определено.

Пусть β действует тождественно на переменные $y_{k\ell}$, а значения переменных u_k преобразует в $\overline{u}_k = 1 - u_k,\ k = 1,\dots,K$. Тогда каждое допустимое решение $(\mathbf{Y},\mathbf{u})$ простейшей задачи размещения производства (3.11)–(3.14) отображается в допустимое решение задачи Π_2' с достаточно большим значением целевой функции $g'(\mathbf{Y},\overline{\mathbf{u}}) = -f_{\text{splp}}(\mathbf{Y},\mathbf{u}) > -\lambda$. Если пара $(\mathbf{Y},\overline{\mathbf{u}})$ является допустимой для задачи Π_2', но при этом $(\mathbf{Y},\mathbf{u})$ не является допустимым решением простейшей задачи размещения производства (3.11)–(3.14), то $g'(\mathbf{Y},\overline{\mathbf{u}}) \le -f_{\text{splp}}(\mathbf{Y},\mathbf{u}) - \lambda$, так как по крайней мере одно из неравенств (3.12) нарушается при подстановке решения $(\mathbf{Y},\mathbf{u})$.

Согласно следствию 3.1 ЗОР для задачи Π_2' может быть решена за полиномиальное время, поэтому из утверждения 3.2 следует полиномиальная разрешимость ЗОР для Π_1'. $\square$

3.2.4. Анализ задач булевого линейного программирования с использованием гиперграфов

Отправной точкой для построения всех сводимостей, рассмотренных выше, была теорема 3.1, полученная с использованием преобразования ЗОР в задачу о независимом множестве наибольшего веса в двудольном графе. С целью обобщения этого результата перейдем от графов к гиперграфам.

Гиперграф $H = (V, E)$ задается конечным непустым множеством вершин V и множеством ребер E, где каждое ребро $e \in E$ является подмножеством в V. Множество вершин гиперграфа $S \subseteq V$ называется *независимым*, если ни одно из ребер $e \in E$ не содержится в S. Задача о независимом множестве наибольшего веса в гиперграфе $H = (V, E)$ с весами вершин $w(v) \in \mathbb{Q},\ v \in V$ состоит в отыскании независимого множества S с наибольшим весом $w(S) = \sum_{v \in S} w(v)$.

Обобщением двудольных графов являются *2-раскрашиваемые гиперграфы*, т. е. такие гиперграфы, для которых существует разбиение множества вершин V на два непересекающихся независимых подмножества (цветовых класса) C_1 и C_2. Разбиение $V = C_1 \cup C_2$ при $C_1 \cap C_2 = \emptyset$ называется 2-раскраской гиперграфа H.

Для задачи булевого линейного программирования (3.3)-(3.5), как и ранее в случае задачи упаковки множества, введем обозначение $U_i = \{j \mid q_{ij} \neq 0,\ j = 1, \ldots, n\}$, $i = 1, \ldots, m$. В дальнейшем будем предполагать, что по крайней мере одно из подмножеств U_i содержит два или более элемента (в противном случае задача решается тривиально). Следующий результат является обобщением теоремы 3.1.

Теорема 3.2. *ЗОР для задачи булевого линейного программирования (3.3)-(3.5) сводится к задаче о независимом множестве в 2-раскрашиваемом гиперграфе, где 2-раскраска является частью вход-*

ных данных. Каждое ребро в получаемом гиперграфе содержит не более чем $U_{\max}$ *вершин, где* $U_{\max} = \max_{i=1,\dots,m} |U_i|$*, и трудоемкость данной сводимости есть* $O(m(2^{U_{\max}} + n))$.

Доказательство. Пусть $\mathbf{p}^1$ и $\mathbf{p}^2$ – родительские решения в задаче булевого линейного программирования. Обозначим число различий между этими решениями $|D(\mathbf{p}^1, \mathbf{p}^2)|$ через d и построим гиперграф H на множестве из $2d$ вершин, сопоставляя каждой переменной $x_j, j \in D(\mathbf{p}^1, \mathbf{p}^2)$ пару вершин v_j, v_{n+j}.

Для каждого линейного ограничения при $i = 1, \dots, m$ осуществим перебор всех возможных комбинаций значений булевых переменных из $D(\mathbf{p}^1, \mathbf{p}^2)$, встречающихся в данном ограничении:

$$\{\mathbf{x} \in \{0,1\}^n : x_j = 0 \ \forall j \notin U_i \cap D(\mathbf{p}^1, \mathbf{p}^2)\}.$$

Пусть $\mathbf{x}^{ik}, k = 1, \dots, 2^{|U_i \cap D(\mathbf{p}^1, \mathbf{p}^2)|}$, обозначает k-тый вектор в этом множестве. Для комбинации с номером k в случае нарушения ограничения i из (3.4), т. е. при

$$\sum_{j \in U_i \cap D(\mathbf{p}^1, \mathbf{p}^2)} q_{ij} x_j^{ik} + \sum_{j \in U_i \setminus D(\mathbf{p}^1, \mathbf{p}^2)} q_{ij} p_j^1 > b_i,$$

добавляем в гиперграф ребро

$$e_{ik} = \{v_j : x_j^{ik} = 1,\ j \in U_i \cap D(\mathbf{p}^1, \mathbf{p}^2)\} \cup \{v_{j+n} : x_j^{ik} = 0,\ j \in U_i \cap D(\mathbf{p}^1, \mathbf{p}^2)\}.$$

(Заметим, что ребро e_{ik} содержит не более $|U_i|$ вершин.) Кроме того, для каждого $j \in D(\mathbf{p}^1, \mathbf{p}^2)$ добавляем d ребер вида $\{v_j, v_{n+j}\}$ – это гарантирует, что обе вершины v_j и v_{n+j} не могут входить в независимое множество одновременно.

Если $\mathbf{x}$ – допустимое решение ЗОР для задачи (3.3)-(3.5), то множество вершин

$$S(\mathbf{x}) = \{v_j : x_j = 1, j \in D(\mathbf{p}^1, \mathbf{p}^2)\} \cup \{v_{j+n} : x_j = 0, j \in D(\mathbf{p}^1, \mathbf{p}^2)\}$$

независимо в H. При заданном множестве вершин S соответствующий ему вектор $\mathbf{x}(S)$ может быть построен назначением $\mathbf{x}(S)_j = 1$, если $v_j \in S$, $j \in D(\mathbf{p}^1, \mathbf{p}^2)$ или $p_j^1 = p_j^2 = 1$; в противном случае $\mathbf{x}(S)_j = 0$. Тогда для любого независимого множества S, состоящего из d вершин, вектор $\mathbf{x}(S)$ является допустимым в задаче булевого линейного программирования.

Вершинам гиперграфа приписываются следующие веса:

$$w(v_j) = c_j + \lambda,\ w(v_{n+j}) = \lambda,\ j \in D(\mathbf{p}^1, \mathbf{p}^2),$$

где $\lambda > 2\sum_{j \in D(\mathbf{p}^1, \mathbf{p}^2)} |c_j|$.

В построенном гиперграфе всякое независимое множество наибольшего веса S^* содержит либо v_j, либо v_{n+j} для любого $j \in D(\mathbf{p}^1, \mathbf{p}^2)$. В самом деле ЗОР имеет допустимое решение, и это решение соответствует независимому множеству с весом не менее λd. Однако если независимое множество не содержит ни v_j, ни v_{n+j} для некоторого j, то его вес меньше, чем $\lambda d - \lambda/2$.

Таким образом, оптимальное независимое множество S^* соответствует допустимому вектору $\mathbf{x}(S^*)$ со значением целевой функции

$$\mathbf{cx}(S^*) = \sum_{j \in S^*,\ j \le n} c_j + \sum_{j \notin D(\mathbf{p}^1, \mathbf{p}^2)} c_j p_j^1 = w(S^*) - \lambda d + \sum_{j \notin D(\mathbf{p}^1, \mathbf{p}^2)} c_j p_j^1.$$

При действии обратного к $\mathbf{x}(S)$ отображения $S(\mathbf{x})$ всякий допустимый вектор $\mathbf{x}$ преобразуется в независимое множество с весом

$$w(S(\mathbf{x})) = \mathbf{cx} + \lambda d - \sum_{j \notin D(\mathbf{p}^1, \mathbf{p}^2)} c_j p_j^1,$$

поэтому $\mathbf{x}(S^*)$ является оптимальным решением ЗОР. □

Заметим, что если ребро $e \in H$ состоит из одной вершины, $e = \{v\}$, то v не может входить в какое-либо независимое множество. Все такие вершины следует исключить из гиперграфа H, построенного в сводимости из теоремы 3.2. Обозначим полученный гиперграф через H'. Если $U_{\max} \le 2$, то H' представляет собой простой граф с не более чем $2d$ вершинами. Таким

образом, по теореме 3.2 ЗОР сводится к решению задачи о независимом множестве в двудольном графе H', которая разрешима за время $O(d^3)$. Используя этот факт, можно обобщить результат из [122].

Следствие 3.3. *ЗОР для задачи булевого линейного программирования, где в каждое неравенство входит не более двух переменных, разрешима за время* $O(|D(\mathbf{p}^1, \mathbf{p}^2)|^3 + mn)$.

Уместен вопрос о целесообразности предоставления во входных данных задачи о независимом множестве вместе с гиперграфом H также и некоторой его 2-раскраски (см. формулировку теоремы 3.2). В частном случае, когда H является простым двудольным графом, разбиение множества его вершин на доли (2-раскраска) осуществляется с линейной трудоемкостью. В общем случае, однако, отыскание 2-раскраски может иметь значительную трудоемкость. Известно, к примеру, что в случае, если каждое ребро гиперграфа состоит из 4-х вершин, отыскание 2-раскрашивания для 2-раскрашиваемого гиперграфа является NP-трудной задачей [206].

3.2.5. Заключение

В настоящем разделе показано, что для целого ряда NP-трудных задач комбинаторной оптимизации задача оптимальной рекомбинации может быть решена за полиномиальное время. При построении эффективных алгоритмов решения задач оптимальной рекомбинации нами были использованы известные сводимости между задачами комбинаторной оптимизации.

Остается открытым вопрос о соотношении трудоемкости решения задачи оптимальной рекомбинации с общей трудоемкостью ЭА, например, в терминах среднего времени первого достижения оптимального решения. Наиболее перспективным представляется экспериментальное исследование этого вопроса, однако теоретические подходы, предложенные в [157] и [260] могут также оказаться плодотворными.

3.3. NP-трудные случаи

3.3.1. Задачи булевого линейного программирования с ограниченным числом переменных в каждом неравенстве

Как было установлено выше, оптимальная рекомбинация для задач булевого линейного программирования связана с задачей о независимом множестве в гиперграфах при заданной 2-раскраске. Последняя задача является NP-трудной, как показывает следующее

Утверждение 3.3. *Задача отыскания наибольшего независимого множества в гиперграфе, где ребра имеют мощность, равную 3, является NP-трудной в сильном смысле, даже при заданной 2-раскраске.*

Доказательство. Сведем к рассматриваемой задаче NP-трудную в сильном смысле задачу о независимом множестве в простом графе [27]. Пусть дан граф $G = (V, E)$ с множеством вершин $V = \{v_1, \ldots, v_n\}$ и множеством ребер E. Рассмотрим гиперграф $H = (V', E')$ на множестве вершин $V' = \{v_1, \ldots, v_{2n}\}$, где для каждого ребра $e = \{v_i, v_j\} \in E$ имеется n ребер вида $\{v_i, v_j, v_{n+k}\},\ k = 1, \ldots, n$, в E'. Для этого графа 2-раскраска образуется цветовыми классами $C_1 = V$ и $C_2 = \{v_{n+1}, \ldots, v_{2n}\}$. Любое из наибольших независимых множеств в этом гиперграфе состоит из множества вершин $\{v_{n+1}, \ldots, v_{2n}\}$ в объединении с некоторым наибольшим независимым множеством S^* графа G. Следовательно, любое из наибольших независимых множеств для H эффективно отображается в некоторое наибольшее независимое множество для G, вычисление которого представляет собой NP-трудную задачу. □

Задача отыскания наибольшего независимого множества в гиперграфе $H = (V, E)$ может быть сформулирована как задача булевого программирования

$$\max \left\{ \sum_{j=1}^{n} x_j : \mathbf{Qx} \leq \mathbf{b},\ \mathbf{x} \in \{0,1\}^n \right\}, \tag{3.22}$$

где $m = |E|, n = |V|$, $b_i = |e_i| - 1,\ i = 1, \ldots, m$, и $q_{ij} = 1$ в том и только том случае, когда $v_j \in e_i$, иначе $q_{ij} = 0$. В частном случае, когда H является 2-раскрашиваемым, векторы $\mathbf{p}^1$ и $\mathbf{p}^2$ могут быть заданы как индикаторы цветовых классов C_1 и C_2 какой-либо 2-раскраски. В таком случае $D(\mathbf{p}^1, \mathbf{p}^2) = \{1, \ldots, n\}$ и ЗОР для задачи булевого линейного программирования (3.22) оказывается эквивалентной задаче о независимом множестве в гиперграфе H с заданной 2-раскраской. Ввиду утверждения 3.3 это влечет

Следствие 3.4. *ЗОР для задачи булевого линейного программирования NP-трудна в сильном смысле, даже если* $|U_i| = 3$ *для всех* $i = 1, \ldots, m$, $c_j = 1$ *для всех* $j = 1, \ldots, n$ *и матрица* $\mathbf{Q}$ *состоит из нулей и единиц.*

3.3.2. Задачи об одномерном булевом рюкзаке и одномерной упаковке в контейнеры

Ниже будет рассмотрена сложность ЗОР для задачи об одномерном булевом рюкзаке (см. раздел 1.1). В постановке булевого линейного программирования эта задача имеет вид

$$\max \{\mathbf{cx} : \mathbf{ax} \leq A, \mathbf{x} \in \{0,1\}^n\}, \tag{3.23}$$

где $\mathbf{c} = (c_1, \ldots, c_n)$, $\mathbf{a} = (a_1, \ldots, a_n)$, $a_j \geq 0, c_j \geq 0, j = 1, \ldots, n$, и $A \geq 0$ целочислены.

Кроме того, здесь будет рассмотрена сложность ЗОР для следующей задачи об *одномерной упаковке в контейнеры*. Задан размер контейнера A и k целых чисел $a_1, \ldots, a_k$, представляющих собой размеры предметов, $a_i \leq A,\ i = 1, \ldots, k$. Требуется разместить предметы в минимальное число контейнеров, чтобы сумма размеров предметов в каждом контейнере не превышала величины A.

Сформулированная задача может быть представлена в виде задачи целочисленного линейного программирования различными способами [56].

Рассмотрим один из них.

Пусть булева переменная y_j является индикатором использования контейнера с номером j, $j = 1, \ldots, k$, а булева переменная x_{ij} – индикатором упаковки предмета i в контейнер j при $i, j = 1, \ldots, k$. Требуется найти

$$\min \sum_{j=1}^{k} y_j \tag{3.24}$$

при условиях

$$\sum_{j=1}^{k} x_{ij} = 1, \quad i = 1, \ldots, k, \tag{3.25}$$

$$\sum_{i=1}^{k} a_i x_{ij} \leq A, \quad j = 1, 2, \ldots, k, \tag{3.26}$$

$$y_j \geq x_{ij}, \quad i = 1, \ldots, k, \ j = 1, \ldots, k, \tag{3.27}$$

$$x_{ij}, \ y_j \in \{0, 1\}, \quad i = 1, \ldots, k, \ j = 1, 2, \ldots, k. \tag{3.28}$$

Заметим, что для кодировки решений данной задачи достаточно указать лишь *матрицу назначений* (x_{ij}), так как вектор $(y_1, \ldots, y_k)$, соответствующий такой матрице, вычисляется однозначно. Далее будем предполагать, что решения задачи об одномерной упаковке в контейнеры кодируются посредством $k \times k$-матрицы назначений.

Для анализа сложности ЗОР для задач об одномерном булевом рюкзаке и одномерной упаковке в контейнеры нам потребуется одно свойство задачи о разбиении предметов, упоминаемое в работе П. Шурман и Г. Воегингера [283], формулируемое здесь как

Лемма 3.1. *Следующий частный случай задачи о разбиении предметов является NP-полным: дано* $2m$ *натуральных чисел* α_j, $j = 1, \ldots, 2m$, *та-*

ких что

$$\frac{B}{m+1} < \alpha_j < \frac{B}{m-1}, \quad j = 1, \ldots, 2m, \tag{3.29}$$

где $B = \sum_{j=1}^{2m} \alpha_j/2$, *и требуется определить, существует ли такой вектор* $\mathbf{x} \in \{0,1\}^{2m}$, *что*

$$\sum_{j=1}^{2m} \alpha_j x_j = B. \tag{3.30}$$

Доказательство. NP-полнота данного частного случая задачи о разбиении предметов может быть показана с помощью сводимости к нему следующей NP-полной модификации задачи о разбиении предметов [27]: дано множество из $2m$ натуральных чисел $\alpha'_j,\ j = 1, \ldots, 2m$, и требуется распознать существование такого $\mathbf{x} \in \{0,1\}^{2m}$, что

$$\sum_{j=1}^{2m} x_j = m \quad \text{и} \quad \sum_{j=1}^{2m} \alpha'_j x_j = \frac{1}{2} \sum_{j=1}^{2m} \alpha'_j. \tag{3.31}$$

Сводимость состоит в назначении $\alpha_i = \alpha'_i + M,\ i = 1, \ldots, 2m$, при достаточно большом целочисленном значении M, например, $M = 2m \cdot \max\{\alpha'_j : j = 1, \ldots, 2m\}$. Непосредственно проверяется выполнение (3.29), а также эквивалентность (3.31) и (3.30) при данном наборе α_i. □

В следующей теореме наряду с задачей оптимальной рекомбинации рассматривается ослабленный ее вариант в связи с тем, что NP-трудность ослабленной ЗОР для задачи об одномерном рюкзаке будет представлять интерес в разделе 5.2.

Теорема 3.3. *ЗОР и ослабленная ЗОР для одномерных задач о булевом рюкзаке и об упаковке в контейнеры являются NP-трудными.*

Доказательство. 1. Рассмотрим ЗОР для задачи о булевом рюкзаке (3.23). Обоснование NP-трудности данной ЗОР основано на полиномиальном сведении к ней (по Тьюрингу) NP-полного частного случая задачи о разбиении предметов с условием (3.29). Будем предполагать, что $m > 2$.

Заметим, что если задача о разбиении предметов с условием (3.29) имеет ответ «да», то существует вектор $\mathbf{x}' \in \{0,1\}^{2m}$, такой что $\sum_{i=1}^{2m} \alpha_i x'_i = B$, и так как $B/(m+1) < \alpha_i < B/(m-1),\ i = 1,\ldots,2m$, то данный вектор содержит ровно m единиц. Поскольку $m > 2$, вектор $\mathbf{x}'$ содержит менее $2m-2$ единиц. С другой стороны, если в задаче о разбиении предметов ответ «нет», то вектора указанного вида не существует.

При сведении задачи о разбиении предметов к ЗОР для задачи о рюкзаке выполняется перебор полиномиального числа различных пар родительских решений (и соответствующих им ЗОР). Полагаем $n = 2m$, $A = B$ и $c_j = a_j = \alpha_j,\ j = 1,\ldots,n$, и осуществляем перебор всех из $\binom{2m}{2}$ возможных фиксаций с нулевым значением каждой пары переменных с индексами $\{i_\ell, j_\ell\},\ \ell = 1,\ldots,\binom{2m}{2}$. Для фиксации переменных с номерами i_ℓ, j_ℓ назначаем $p^1_{i_\ell} = p^2_{j_\ell} = 0$ и заполняем остальные позиции $j \notin \{i_\ell, j_\ell\}$ так, чтобы выполнялись равенства $p^1_j + p^2_j = 1$ и в каждом из родительских решений было бы по $m-1$ единице (родительские решения будут допустимыми, так как $a_j < A/(m-1),\ j = 1,\ldots,n$). Лучшее из оптимальных значений целевой функции для построенных ЗОР будет равно A в том и только том случае, когда в задаче о разбиении предметов ответ «да». Отсюда следует NP-трудность ЗОР для задачи о рюкзаке.

NP-трудность ослабленной ЗОР доказывается с помощью той же сводимости ввиду того, что множества допустимых решений получаемых ЗОР и их ослабленных вариантов совпадают.

2. Доказательство NP-трудности ЗОР для задачи об одномерной упаковке в контейнеры основывается на аналогичной (но более трудоемкой) сводимости по Тьюрингу частного случая задачи о разбиении предметов с условием (3.29). При этом полагаем $k = 2m$, $A = B$, и $a_i = \alpha_i,\ i = 1,\ldots,k$. Далее в доказательстве предполагается, что $m > 4$.

При сведении осуществляем перебор полиномиального числа различных пар родительских решений, выбирая их таким образом, чтобы $2m-4$

предмета в решении-потомке заведомо оказывались бы в первых двух контейнерах, причем два выделенных предмета в обязательном порядке помещались бы во второй контейнер, а четыре предмета могли бы оказаться либо в первом, либо в третьем контейнере. Опишем сводимость подробнее.

Пусть подобно первой части доказательства осуществляется перебор всех $\binom{2m}{2}$ пар предметов $\{i_\ell, i'_\ell\}$, $\ell = 1, \ldots, \binom{2m}{2}$, с целью фиксации нулевых значений в соответствующих переменных из первого столбца $\{x_{i_\ell,1}, x_{i'_\ell,1}\}$. Для каждой такой пары предметов далее осуществляется перебор всех $\binom{2m-2}{2}$ пар $\{u_r, u'_r\}$, $r = 1, \ldots, \binom{2m-2}{2}$, из оставшихся предметов. После выбора $\{i_\ell, i'_\ell\}$ и $\{u_r, u'_r\}$ осуществляется перебор всех $\binom{2m-4}{2}$ пар $\{v_s, v'_s\}$, $s = 1, \ldots, \binom{2m-4}{2}$ из оставшихся предметов.

Для того чтобы при заданных ℓ, r и s в решении-потомке предметы $\{i_\ell, i'_\ell\}$ находились во втором контейнере, а предметы u_r, u'_r, v_s, v'_s могли оказаться только в первом или третьем контейнере, определим пару родительских решений $\mathbf{p}^1 = (p^1_{ij})$ и $\mathbf{p}^2 = (p^2_{ij})$ следующим образом.

В первом столбце родительских решений положим

$$p^1_{i_\ell,1} = p^1_{i'_\ell,1} = p^2_{i_\ell,1} = p^2_{i'_\ell,1} = 0,$$

$$p^1_{u_r,1} = p^1_{u'_r,1} = 1,\ p^2_{u_r,1} = p^2_{u'_r,1} = 0,$$

$$p^1_{v_s,1} = p^1_{v'_s,1} = 0,\ p^2_{v_s,1} = p^2_{v'_s,1} = 1$$

и заполним остальные позиции $i \notin \{i_\ell, i'_\ell, u_r, u'_r, v_s, v'_s\}$ так, чтобы выполнялись равенства $p^1_{i,1} + p^2_{i,1} = 1$ и при этом в каждом из родительских решений в первом столбце было бы по $m-1$ единице. Такие родительские решения удовлетворяют ограничению (3.26) для контейнера $j = 1$, так как $a_i < A/(m-1)$, $i = 1, \ldots, k$.

Пусть второй столбец в каждом из родительских решений заполнен так же, как первый столбец *другого* родительского решения, за исключением следующих шести компонент. Во-первых, два единичных элемента в строках v_s и v'_s родительского решения $\mathbf{p}^1$ вместо столбца $j = 2$ поме-

щаются в столбец $j = 3$. Во-вторых, два единичных элемента в строках u_r и u'_r родительского решения $\mathbf{p}^2$ вместо столбца $j = 2$ помещаются в столбец $j = 3$. В-третьих, в столбец $j = 2$ обоих родительских решений добавляются единичные элементы в строках с номерами i_ℓ, i'_ℓ.

Таким образом, в каждом из родительских решений во втором столбце содержится по $m - 1$ единице, поэтому условие (3.26) для контейнера $j = 2$ в обоих родительских решениях выполняется так же, как и в случае $j = 1$. Для контейнера $j = 3$ данное условие верно, так как $a_i < A/4$, $i = 1, \ldots, k$ при $m > 4$. Заметим, что в допустимых решениях ЗОР, соответствующей набору индексов ℓ, r, s, предметы i_ℓ, i'_ℓ попадут во второй контейнер в обязательном порядке, а предметы u_r, u'_r, v_s и v'_s окажутся либо в первом, либо в третьем контейнере.

Заметим, что если в задаче о разбиении предметов ответ «да», то в одной из построенных ЗОР значение целевой функции будет равно 2. Действительно, в таком случае в векторе $\mathbf{x}'$, удовлетворяющем условию (3.30), существуют две позиции $\hat{i}, \bar{i}$, такие что $x'_{\hat{i}} = x'_{\bar{i}} = 0$. Кроме того, существуют четыре позиции $\hat{u}, \bar{u}, \hat{v}$ и $\bar{v}$, такие что $x'_{\hat{u}} = x'_{\bar{u}} = x'_{\hat{v}} = x'_{\bar{v}} = 1$, так как данный вектор содержит не менее m единиц. Соответствующая ЗОР при $\{i_\ell, i'_\ell\} = \{\hat{i}, \bar{i}\}$, $\{u_r, u'_r\} = \{\hat{u}, \bar{u}\}$ и $\{v_s, v'_s\} = \{\hat{v}, \bar{v}\}$ имеет допустимое решение (x'_{ij}), где первый столбец совпадает с вектором $\mathbf{x}'$, второй столбец содержит элементы $x'_{i2} = 1 - x'_i$, $i = 1, \ldots, k$, а все прочие столбцы – нулевые.

С другой стороны, если оптимальное решение x^*_{ij} в одной из построенных ЗОР имеет значение целевой функции, равное 2, то, полагая $x_i = x^*_{i1}$, $i = 1, \ldots, k$, получаем равенство (3.30).

Таким образом, построена полиномиальная сводимость по Тьюрингу NP-трудного по лемме 3.1 частного случая задачи о разбиении предметов к ЗОР для одномерной задачи об упаковке в контейнеры.

NP-трудность ослабленной ЗОР, как и в случае задачи о рюкзаке,

следует из совпадения множеств допустимых решений получаемых ЗОР и их ослабленных вариантов. □

3.3.3. Задачи о наименьшем покрытии и размещении производства

Следующий пример NP-трудной ЗОР порождается задачей о наименьшем покрытии, которая может рассматриваться как частный случай задачи (3.3)-(3.5):

$$\min\{\mathbf{cx} : \mathbf{Qx} \geq \mathbf{e},\ \mathbf{x} \in \{0,1\}^n\}, \tag{3.32}$$

где $\mathbf{Q}$ – матрица из нулей и единиц размерности $m \times n$; $\mathbf{c} = (c_1, \ldots, c_n)$, $c_j \geq 0,\ j = 1, \ldots, n$. Будем предполагать двоичное представление решений с помощью вектора $\mathbf{x}$. При заданной индивидуальной задаче о наименьшем покрытии может быть построена новая индивидуальная задача о наименьшем покрытии, содержащая удвоенный набор столбцов в матрице, т. е. $\mathbf{Q}' = (\mathbf{QQ})$ и удвоенный вектор $\mathbf{c}' = (c_1, \ldots, c_n, c_1, \ldots, c_n)$. Индивидуальная задача NP-трудной задачи о наименьшем покрытии (3.32) эквивалентна ЗОР для индивидуальной задачи о наименьшем покрытии, где вход представляет собой $(m \times 2n)$-матрицу $\mathbf{Q}'$, $2n$-вектор $\mathbf{c}'$ и пару допустимых решений $\mathbf{p}^1, \mathbf{p}^2$, где $p^1_j = 1, p^2_j = 0$ для $j = 1, \ldots, n$ и $p^1_j = 0, p^2_j = 1$ для $j = n+1, \ldots, 2n$. Отсюда следует, что ЗОР для задачи о наименьшем покрытии является NP-трудной.

Интересно отметить, что в некоторых случаях ЗОР может быть более сложной, чем исходная задача (в предположении P$\neq$ NP). Проиллюстрируем это на примере задачи о наименьшем покрытии. Рассмотрим сужение задачи, при котором $q_{i,1} = 1, i = 1, \ldots, m;\ c_1 = 0$. Очевидно, вектор $\mathbf{x} = (1, 0, 0, \ldots, 0)$ является оптимальным решением для этого класса индивидуальных задач. Однако уже в частном случае, где $p^1_1 = p^2_1 = 0$, ЗОР оказывается NP-трудной задачей.

Задача о наименьшем покрытии эффективно сводится к простейшей задаче размещения производства (1.1) – (1.2) – см., например, [12, 232]. При этом размерности m и n в обеих задачах имеют одинаковые значения, $C_i = c_i$ для $i = 1, \ldots, n$, и

$$c_{ij} = \begin{cases} \sum_{k=1}^{n} c_k + 1, & \text{если } \ q_{ij} = 0, \\ 0, & \text{если } \ q_{ij} = 1, \end{cases} \quad \text{при } \ i = 1, \ldots, n, \ j = 1, \ldots, m.$$

Вектор $\mathbf{x}^*$ оптимального решения данной задачи будет оптимальным решением соответствующей задачи о наименьшем покрытии. Таким образом, если представление решений для простейшей задачи размещения производства определяется одним вектором $\mathbf{x}$, то указанная сводимость удовлетворяет условиям утверждения 3.2. При этом множество получаемых решений простейшей задачи размещения производства $\beta(\mathrm{Sol}_1(I))$ характеризуется пороговым значением целевой функции $f_{\mathrm{splp}}(\mathbf{x}) < \sum_{j=1}^{n} c_j + 1$, гарантирующим выполнение всех ограничений задачи о наименьшем покрытии. Таким образом, к рассматриваемой ЗОР полиномиально сводится NP-трудная задача, а значит, имеет место

Утверждение 3.4. *ЗОР для простейшей задачи размещения производства (1.1) – (1.2) является NP-трудной задачей.*

Известная задача о p-медиане может быть сформулирована как модификация простейшей задачи размещения производства: достаточно предположить, что все $C_k = 0$, и добавить в задачу ограничение $\sum_{i=1}^{n} x_i = p$, где $p \in \{1, \ldots, n\}$ – параметр, являющийся частью входных данных.

Утверждение 3.5. *ЗОР для задачи о p-медиане с кодировкой решений вектором открытых предприятий является NP-трудной задачей.*

Доказательство. В работе [110] предложена полиномиальная сводимость NP-трудной задачи о разбиении графа к классу задач о p-медиане, где $p = n/2$ и при этом n является четным. Таким образом, указанный

класс задач о p-медиане также является NP-трудным. Рассмотрим ЗОР для задачи о p-медиане из этого класса с представлением решений вектором открытых предприятий $\mathbf{x}$. Пусть родительские решения имеют вид $\mathbf{p}^1 = (1, \ldots, 1, 0, \ldots, 0)$, $\mathbf{p}^2 = (0, \ldots, 0, 1, \ldots, 1)$ и содержат по $n/2$ единиц. Получаемая ЗОР эквивалентна порождающей ее задаче о p-медиане. □

3.3.4. Задача максимальной выполнимости логической формулы

Задача максимальной выполнимости логической формулы – одна из классических NP-трудных задач комбинаторной оптимизации [27]. Дана логическая формула в виде конъюнкции скобок $\mathbf{c}_1, \ldots, \mathbf{c}_M$, где каждая скобка представляет собой дизъюнкцию логических переменных (т. е. переменных, принимающих значения «истина» или «ложь») или их отрицаний. Требуется максимизировать число выполненных скобок.

Пусть n – число логических переменных, входящих в формулу. Наиболее естественное и компактное представление решений для данной задачи состоит в использовании вектора $\mathbf{x} \in \{0, 1\}^n$, каждая компонента которого сопоставляется одной из логических переменных и равна 0, если логическая переменная имеет значение «ложь», или 1, если логическая переменная имеет значение «истина». ЗОР для задачи выполнимости логической формулы с таким представлением решений представляет собой NP-трудную задачу. Для доказательства этого рассмотрим пару родительских решений, где $p_j^1 + p_j^2 = 1$ для всех $j = 1, \ldots, n$, задающую индивидуальную задачу оптимальной рекомбинации. Эта задача эквивалентна исходной задаче максимальной выполнимости логической формулы.

3.3.5. Задача коммивояжера

Рассмотрим задачу коммивояжера (см. раздел 1.1) с точки зрения сложности ЗОР. В общем случае допустимое решение, т. е. маршрут ком-

мивояжера – гамильтонов контур в полном ориентированном графе без петель и кратных дуг с множеством вершин V и множеством дуг $\mathcal{A}$, где длина дуги $(i,j) \in \mathcal{A}$, равна c_{ij}. В симметрическом же случае направление обхода не имеет значения, поэтому маршрутом коммивояжера является гамильтонов цикл в полном неориентированном графе G с тем же множеством вершин V и множеством ребер E, где длина ребра $\{i,j\}$ есть $c_{ij} = c_{ji}$. Далее обозначим число вершин $|V|$ через $\hat{n}$.

Согласно предположению, сделанному в разделе 1.1, элемент множества $\mathrm{Sol}(I)$ в общем случае ЗК представляет собой строку, где последовательно записаны компоненты матрицы перестановки, отвечающей маршруту коммивояжера. Заметим, что в симметрическом случае для кодировки решения нет надобности в элементах, находящихся над главной диагональю этой матрицы. Таким образом, в общем случае $n = \hat{n}^2$, а в симметрическом случае $n = \hat{n}(\hat{n}+1)/2$. ЗОР для ЗК с данным способом кодировки решений представляет собой задачу поиска кратчайшего маршрута коммивояжера, совпадающего с двумя родительскими допустимыми решениями по тем дугам (или ребрам), по которым проходят оба родительских решения, и не проходящего по дугам (или ребрам), отсутствующим в обоих из них.

Операторы оптимальной рекомбинации для задач, где множество допустимых решений составляют перестановки, применяли в своих работах М. Ягиура и Т. Ибараки [294], К. Котта, Е. Алба и Дж.М. Тройа [149], В. Кук и П. Сеймур [147], Д. Уитли, Д. Хайнс и А. Хоуве [292] и другие авторы. Формулировки вспомогательных задач рекомбинации в [42,147,149,294] несколько отличаются от ЗОР, рассматриваемой здесь. Например, в [147] исследована ослабленная ЗОР, решение которой строится на множестве ребер, входящих хотя бы в одно из родительских решений. В работе [42] для задачи составления расписаний многопродуктового производства с группировкой машин по технологиям предложена задача оптимальной реком-

бинации с использованием недвоичного представления решений, где гены соответствуют позициям в расписании, а их значениями являются номера технологий. Как следует из результата Ю.В. Коваленко [43], данная ЗОР является NP-трудной.

Ниже с использованием результатов А. Итаи, Х. Пападимитриу и Дж. Шварцфитера [217] показана NP-трудность задачи оптимальной рекомбинации в симметрическом случае. Здесь также доказана NP-трудность оптимальной рекомбинации в общем случае с использованием известной идеи преобразования задачи вершинного покрытия в задачу коммивояжера [27]. В завершение предложены сводимости рассматриваемой задачи к задаче коммивояжера на графах со степенью вершин не более чем 4 в симметрическом и не более чем 3 – в общем случае. Решение таких задач может быть получено, например, алгоритмами Д. Эппштейна [171], имеющими оценки трудоемкости существенно ниже, чем известная оценка $O(\hat{n}^2 2^{\hat{n}})$ алгоритма динамического программирования М. Хельда и Р. Карпа [211].

Симметрический случай

В [217] доказана NP-полнота задачи проверки свойства гамильтоновости решеточных графов. Напомним, что граф $G' = (V', E')$ с множеством вершин V' и множеством ребер E' называется *решеточным*, если его вершины имеют вид $v = (x_v, y_v)$ и лежат на евклидовой плоскости в точках с целочисленными координатами $(x_v, y_v) \in \mathbb{Z}^2$, причем пара вершин соединена ребром тогда и только тогда, когда евклидово расстояние между ними равно 1. Будем называть ребра, соединяющие вершины в $\mathbb{Z}^2$ с равными первыми координатами, *вертикальными*; ребра же, соединяющие вершины с равными вторыми координатами, – *горизонтальными*.

Будем предполагать, что $|V'| > 4$, граф G' связен и не имеет мостов (в исключаемых из рассмотрения частных случаях проверка гамильтоновости выполнима за полиномиальное время). Построим сводимость задачи

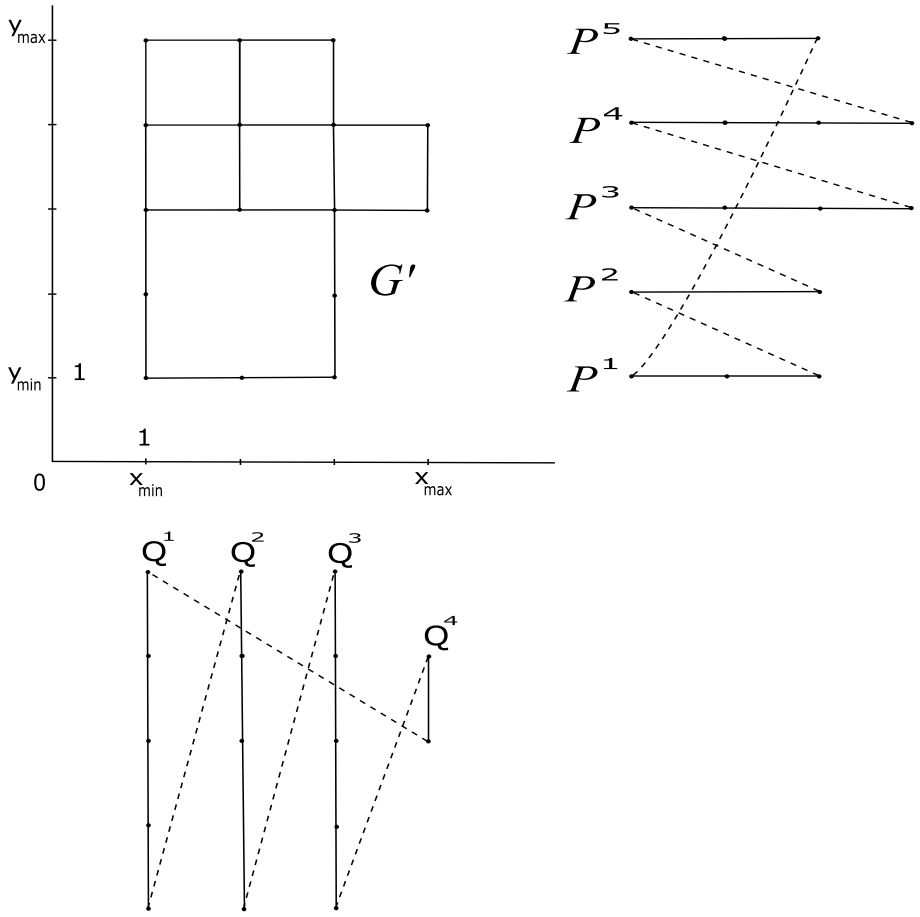

Рис. 3.1. Пример пары родительских маршрутов при сводимости задачи распознавания гамильтоновости решеточного графа к задаче оптимальной рекомбинации в симметрическом случае.

распознавания свойства гамильтоновости решеточного графа G' к задаче оптимальной рекомбинации на некотором взвешенном полном неориентированном графе $G = (V, E)$, где $V = V'$.

Пусть веса ребер c_{ij} в графе G определены следующим образом. Если пара вершин $\{v_i, v_j\}$ соединена ребром в G', то $c_{ij} = 0$. Всем другим ребрам в G приписан вес, равный 1. Рассмотрим следующие два родительских решения задачи коммивояжера на графе G (пример графа G' и двух родительских решений приводится на Рисунке 3.1).

Пусть $y_{\min} = \min_{v \in V'} y_v$, $y_{\max} = \max_{v \in V'} y_v$. Для любого целочисленного $y \in \{y_{\min}, \ldots, y_{\max}\}$ обозначим через P^y горизонтальную цепь, про-

ходящую монотонно по возрастанию координаты x по вершинам $v \in V'$, таким что $y_v = y$. Пусть первый родительский маршрут следует по цепям $P^{y_{\min}}, P^{y_{\min}+1}, \ldots, P^{y_{\max}}$, переходя с правого конца каждой цепи P^y при $y < y_{\max}$ на левый конец цепи P^{y+1}. Заметим, что такие переходы нигде не выполняются по вертикальным ребрам ввиду того, что граф G' не имеет мостов. Для замыкания конструкции в цикл соединим ребром правый конец v_{TR} цепи $P^{y_{\max}}$ с левым концом v_{BL} цепи $P^{y_{\min}}$.

Аналогично построим второй родительский маршрут из вертикальных цепей. Пусть $x_{\min} = \min_{v \in V'} x_v$, $x_{\max} = \max_{v \in V'} x_v$. Для любого целого $x \in \{x_{\min}, \ldots, x_{\max}\}$ обозначим через Q^x вертикальную цепь, проходящую монотонно по y по вершинам $v \in V'$, таким что $x_v = x$. Второй родительский маршрут следует по цепям $Q^{x_{\min}}, Q^{x_{\min}+1}, \ldots, Q^{x_{\max}}$, переходя с нижнего конца каждой цепи Q^x при $x < x_{\max}$ к верхнему концу цепи Q^{x+1}. Указанные переходы нигде не выполняются по горизонтальным ребрам, так как граф G' не имеет мостов. Нижний конец v_{RB} цепи $Q^{x_{\max}}$ соединим с верхним концом v_{LT} цепи $Q^{x_{\min}}$.

Заметим, что первый и второй циклы не имеют общих ребер. Действительно, общие наклонные ребра исключаются, так как V' не может состоять из вершин единичного квадрата ($|V'| > 4$). Горизонтальные ребра принадлежат только первому маршруту за исключением той ситуации, когда $y_{v_{\mathrm{RB}}} = y_{v_{\mathrm{LT}}}$ и замыкающее ребро $\{v_{\mathrm{RB}}, v_{\mathrm{LT}}\}$ второго маршрута расположено горизонтально. Но если бы в этой ситуации и первый родительский маршрут имел ребро $\{v_{\mathrm{RB}}, v_{\mathrm{LT}}\}$, то данное ребро являлось бы мостом графа G'. Следовательно, родительские маршруты по горизонтальным ребрам не могут пересекаться. Также и вертикальные ребра принадлежат только второму маршруту за исключением ситуации, когда $x_{v_{\mathrm{TR}}} = x_{v_{\mathrm{BL}}}$ и замыкающее ребро $\{v_{\mathrm{TR}}, v_{\mathrm{BL}}\}$ первого маршрута расположено вертикально. Но в этой ситуации второй родительский маршрут не может содержать ребро $\{v_{\mathrm{TR}}, v_{\mathrm{BL}}\}$, так как G' не имеет мостов.

Отметим также, что объединение ребер родительских решений содержит в себе E'. Следовательно, любой гамильтонов цикл в графе G' – допустимое решение данной задачи оптимальной рекомбинации. С другой стороны, допустимое решение задачи оптимальной рекомбинации имеет нулевое значение целевой функции тогда и только тогда, когда оно проходит только по ребрам из E'. Следовательно, оптимальное значение целевой функции в рассматриваемой задаче оптимальной рекомбинации равно 0 тогда и только тогда, когда в графе G' существует гамильтонов цикл. Доказана следующая теорема.

Теорема 3.4. *Оптимальная рекомбинация для задачи коммивояжера в симметрическом случае NP-трудна в сильном смысле.*

В [217] также доказана NP-полнота задачи проверки наличия в решеточном графе *гамильтоновой цепи*. Оптимальная рекомбинация для этой задачи состоит в поиске в графе кратчайшей гамильтоновой цепи, совпадающей с двумя родительскими гамильтоновыми цепями по тем ребрам, по которым проходят родительских цепи, и не проходящей по ребрам, отсутствующим в обоих из них. Аналогично теореме 3.4 доказывается

Теорема 3.5. *Оптимальная рекомбинация для задачи отыскания кратчайшей гамильтоновой цепи в графе с заданными произвольно длинами ребер NP-трудна в сильном смысле.*

При доказательстве теоремы 3.5, в отличие от теоремы 3.4, случаи, когда граф G' имеет мосты, не исключаются из рассмотрения. Вместо этого при построении сводимости каждый максимальный подграф без мостов рассматривается отдельно.

Общий случай

В общем случае ЗК задача оптимальной рекомбинации не является обобщением ЗОР для симметрической ЗК. Действительно, даже при

симметрической матрице расстояний (c_{ij}) пара родительских маршрутов, понимаемых как контуры, обусловливает существенно иное множество допустимых решений задачи оптимальной рекомбинации, чем та же пара маршрутов, понимаемых как циклы. Таким образом, общий случай требует отдельного доказательства сложности.

Теорема 3.6. *Оптимальная рекомбинация для задачи коммивояжера в общем случае NP-трудна в сильном смысле.*

Доказательство. Для установления сложности общего случая может быть использован аналог известной сводимости задачи о вершинном покрытии к задаче коммивояжера [27].

Пусть индивидуальная задача о вершинном покрытии задана графом $G' = (V', E')$ и требуется найти вершинное покрытие графа G' минимальной мощности. Будем считать, что вершины множества V' пронумерованы, т. е. $V' = \{v_1, \ldots, v_{\hat{n}}\}$, где $n' = |V'|$, и пусть $m' = |E'|$.

Рассмотрим полный ориентированный граф $G = (V, \mathcal{A})$, множество вершин которого состоит из $|E'|$ *проверяющих покрытие* компонент по 12 вершин в каждой: $V_e = \{(v_i, e, k), (v_j, e, k) : 1 \leq k \leq 6\}$ для каждого $e = \{v_i, v_j\} \in E',\ i < j$. Кроме того, в V входят n' *выбирающих* вершин – обозначим их $a_1, \ldots, a_{n'}$, а также *вспомогательная* вершина $a_{n'+1}$.

Пусть в качестве родительских маршрутов в графе G заданы следующие два контура (пример пары таких контуров приведен на Рисунке 3.2).

1. Каждую проверяющую покрытие компоненту V_e, где $e = \{v_i, v_j\} \in E'$ и $i < j$, первый контур проходит за два раза. В первый раз будут пройдены вершины, относящиеся к v_i, в порядке

$$(v_i, e, 1), \ldots, (v_i, e, 6), \tag{3.33}$$

а во второй раз – вершины, относящиеся к v_j, в порядке

$$(v_j, e, 1), \ldots, (v_j, e, 6). \tag{3.34}$$

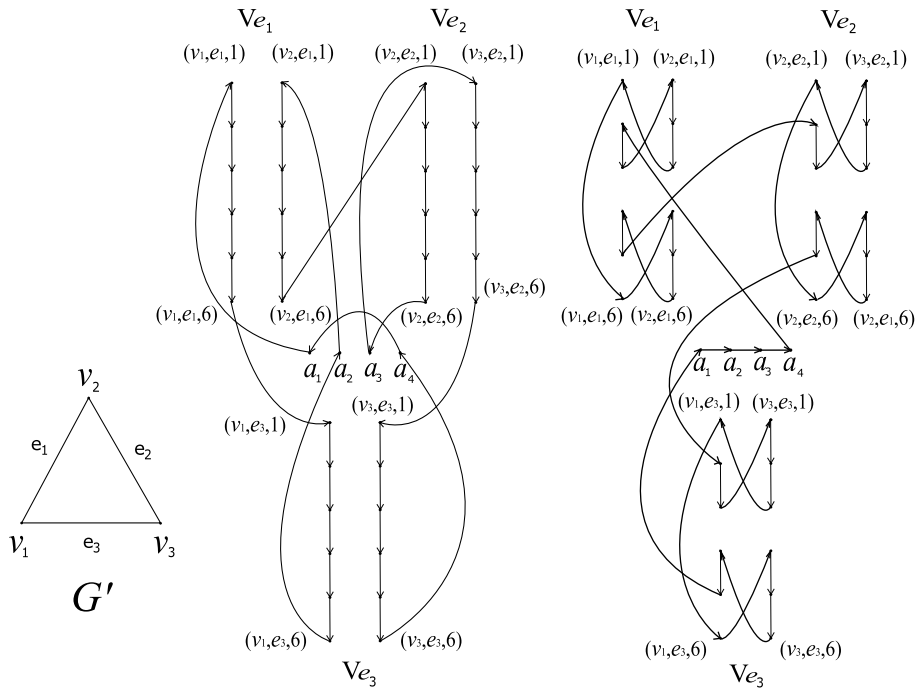

Рис. 3.2. Пара родительских контуров для случая $G' = K_3$. Последовательности обхода ребер: для вершины v^1 $e^{v_1,1} = e_1,\ e^{v_1,2} = e_3$; для вершины v^2 $e^{v_2,1} = e_1,\ e^{v_2,2} = e_2$; для вершины v^3 $e^{v_3,1} = e_2, e^{v_3,2} = e_3$.

2. Второй контур каждую проверяющую покрытие компоненту V_e, где $e = \{v_i, v_j\} \in E'$ и $i < j$, проходит в следующем порядке:

$$(v_i, e, 2), (v_i, e, 3), (v_j, e, 1), (v_j, e, 2), (v_j, e, 3), (v_i, e, 1),$$

$$(v_i, e, 6), (v_j, e, 4), (v_j, e, 5), (v_j, e, 6), (v_i, e, 4), (v_i, e, 5).$$

Между проверяющими покрытие компонентами первый родительский маршрут проходит следующим образом. Для каждой вершины $v \in V'$ произвольно упорядочим ребра графа G', инцидентные v: $e^{v,1}, e^{v,2}, \ldots, e^{v,\deg(v)}$, где $\deg(v)$ – степень вершины v в графе G'. Во всех компонентах проверки покрытия последовательно в выбранном порядке $e^{v,1}, e^{v,2}, \ldots, e^{v,\deg(v)}$ первый маршрут проходит по 6 вершин вида $(v, e, k),\ k = 1, \ldots, 6,\ e \in \{e^{v,1}, e^{v,2}, \ldots, e^{v,\deg(v)}\}$. Таким образом, каждая проверяющая покрытие компонента V_e, где $e = \{u, v\} \in E'$, окажется полностью пройденной двумя участками маршрута по 6 вершин в каждом.

Второй родительский маршрут проходит проверяющие покрытие компоненты в произвольном порядке ребер $V_{e_1}, \ldots, V_{e_{m'}}$, начиная прохождение компоненты V_{e_k} для каждого $e_k = \{v_{i_k}, v_{j_k}\} \in E',\ i_k < j_k,\ k = 1, \ldots, m'$, с $(v_{i_k}, e_k, 2)$ и заканчивая $(v_{i_k}, e_k, 5)$. Таким образом определяется последовательность индексов вершин $i_1, \ldots, i_{m'}$, в которой возможны повторения.

Построенные участки маршрутов замыкаются в гамильтоновы обходы с использованием вершин $a_1, \ldots, a_{n'+1}$: первый замыкается дугами

$$(a_1, (v_1, e^{v_1,1}, 1)),\ ((v_1, e^{v_1,\deg(v_1)}, 6), a_2),$$

$$(a_2, (v_2, e^{v_2,1}, 1)),\ ((v_2, e^{v_2,\deg(v_2)}, 6), a_3),$$

$$\ldots,$$

$$(a_{n'}, (v_{n'}, e^{v_{n'},1}, 1)),\ ((v_{n'}, e^{v_{n'},\deg(v_{n'})}, 6), a_{n'+1}), (a_{n'+1}, a_1),$$

а второй – дугами

$$(a_1, a_2), \ldots, (a_{n'-1}, a_{n'}), (a_{n'}, a_{n'+1}),$$

$$(a_{n'+1}, (v_{i_1}, e_1, 2)),\ ((v_{i_{m'}}, e_{m'}, 5), a_1).$$

Назначим в полном ориентированном графе G единичные веса всем дугам $(a_i, (v_i, e^{v_i,1}, 1)),\ i = 1, \ldots, n'$. Кроме того, назначим веса, равные $n'+1$, всем дугам второго маршрута, соединяющим между собой компоненты $V_{e_1}, \ldots, V_{e_{m'}}$ и дугам $(a_{n'+1}, (v_{i_1}, e_1, 2))$ и $((v_{i_{m'}}, e_{m'}, 5), a_1)$. Всем другим дугам графа G приписывается вес 0.

Заметим, что для любого вершинного покрытия C графа G' множество допустимых решений ЗОР с указанными двумя родительскими маршрутами содержит контур $R(C)$, устроенный следующим образом (контур $R(C)$ для родительских решений из Рисунка 3.2 см. на Рисунке 3.3).

Пусть для каждой $v_i \in C$ в $R(C)$ входят дуги $(a_i, (v_i, e^{v_i,1}, 1))$ и $((v_i, e^{v_i,\deg(v_i)}, 6), a_{i+1})$, а компоненты $V_e,\ e \in \{e^{v_i,1}, e^{v_i,2}, \ldots, e^{v_i,\deg(v_i)}\}$, между собой связывают дуги первого маршрута. Для каждой $v_i \notin C$ в контуре $R(C)$ присутствует дуга (a_i, a_{i+1}). Также в $R(C)$ имеется дуга $(a_{n'+1}, a_1)$.

Пусть $R(C)$ проходит каждую компоненту V_e одним из двух способов:

1. если оба конца ребра e принадлежат C, то $R(C)$ проходит данную компоненту по тем же дугам, что и первый родительский маршрут;

2. если $e = \{u, v\}$, $u \in C$, $v \notin C$, то $R(C)$ проходит вершины данной компоненты в порядке

$$(u, e, 1), (u, e, 2), (u, e, 3),\quad (v, e, 1), \ldots, (v, e, 6),\quad (u, e, 4), (u, e, 5), (u, e, 6).$$

Непосредственная проверка показывает, что в последнем случае обход компоненты V_e не нарушает условий задачи оптимальной рекомбинации.

Выполнение ограничений ЗОР для построенного контура следует из того, что, с одной стороны, все дуги, использованные в $R(C)$, представлены по крайней мере в одном из родительских решений, с другой стороны, в оба родительских маршрута входят лишь дуги вида

$$((u, e, 2), (u, e, 3)),\ ((u, e, 4), (u, e, 5)),\ ((v, e, 1), (v, e, 2)),$$

$$((v, e, 2), (v, e, 3)),\ ((v, e, 4), (v, e, 5)), ((v, e, 5), (v, e, 6))$$

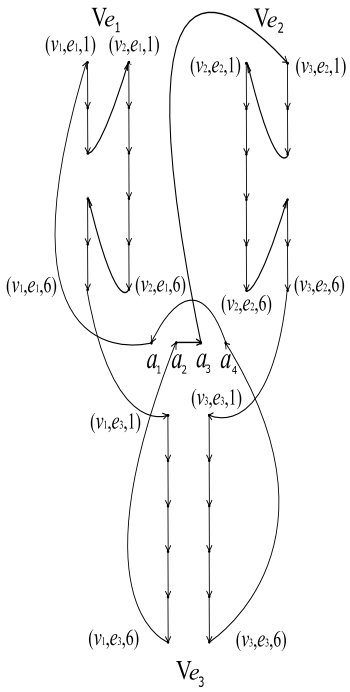

Рис. 3.3. Решение $R(C)$ задачи оптимальной рекомбинации, соответствующее вершинному покрытию $\{v_1, v_3\}$ графа $G' = K_3$.

из компонент V_e, $e = \{u, v\} \in E'$, где u имеет меньший порядковый номер, чем v. Все эти дуги входят в $R(C)$. Длина контура $R(C)$ равна $|C|$.

Поставим в соответствие каждому допустимому решению R построенной ЗОР набор вершин $C(R)$ следующим образом: v_i, $i \in \{1, \ldots, \hat{n}\}$, входит в $C(R)$ тогда и только тогда, когда в R входит дуга $(a_i, (v_i, e^{v_i,1}, 1))$.

Будем рассматривать только решения R со значением целевой функции не более n', что означает отсутствие в них дуг второго маршрута, соединяющих между собой компоненты проверки покрытия, а также дуг $(a_{n'+1}, (v_{i_1}, e_1, 2))$ и $((v_{i_{m'}}, e_{m'}, 5), a_1)$.

Рассмотрим случай, когда дуга $(a_i, (v_i, e^{v_i,1}, 1))$ входит в R. Каждая компонента V_e, где $e = \{v_i, v_j\} \in E'$, в таком случае может быть пройдена одним из двух способов: либо как в первом маршруте (в этом случае, ввиду гамильтоновости R, v_j также будет выбрана в $C(R)$), либо в порядке

$$(v_i, e, 1), (v_i, e, 2), (v_i, e, 3),\ \ (v_j, e, 1), \ldots, (v_j, e, 6),\ \ (v_i, e, 4), (v_i, e, 5), (v_i, e, 6)$$

(в этом случае v_j не будет выбрана в $C(R)$). Поскольку сделано предположение, что дуга $(a_i, (v_i, e^{v_i,1}, 1))$ входит в R, то компоненты проверки покрытия $V_e,\ e \in \{e^{v_i,1}, e^{v_i 2}, \ldots, e^{v_i,\deg(v_i)}\}$, связываются между собой дугами из первого маршрута и, кроме того, в R входит дуга $((v_i, e^{v_i,\deg(v_i)}, 6), a_{i+1})$. Заметим, что суммарная длина дуг контура R равна $|C(R)|$, а множество $C(R)$ – вершинное покрытие в G', так как прохождение маршрута R по каждой компоненте V_e гарантирует покрытие каждого ребра $e \in E'$.

Итак, имеется взаимно-однозначное соответствие, сохраняющее значения целевых функций, между вершинными покрытиями в G' и допустимыми решениями ЗОР с длиной маршрута не более $\hat{n}$. □

Сводимость к задаче коммивояжера на графах с ограниченной степенью вершин

В данном параграфе устанавливается связь рассматриваемых задач оптимальной рекомбинации с модификацией задачи коммивояжера на графе с ограниченной степенью вершин, произвольными положительными длинами ребер и заданным множеством *предписанных* ребер или дуг. В таком графе требуется найти кратчайший гамильтонов цикл или контур, проходящий по всем предписанным ребрам или дугам.

Общий случай. Рассмотрим общий случай задачи, когда в ориентированном полном графе $G = (V, \mathcal{A})$ задано два родительских гамильтоновых маршрута с множествами дуг $\mathcal{A}_1, \mathcal{A}_2$ и требуется решить задачу оптимальной рекомбинации. Данная задача преобразуется в задачу отыскания кратчайшего гамильтонова контура в ориентированном графе $G' = (V', \mathcal{A}')$, полученном из G исключением всех дуг, лежащих в $\mathcal{A} \backslash (\mathcal{A}_1 \cup \mathcal{A}_2)$, и стягиванием каждого пути, содержащегося в обоих родительских маршрутах, в предписанную *псевдодугу* той же длины и направленности, что и данный путь. Длины прочих сохранившихся дуг в G' остаются теми же, что и в G.

Кратчайший гамильтонов контур в G' трансформируется в оптимум задачи оптимальной рекомбинации на G обратной заменой каждой псевдодуги на соответствующий ей путь в оптимальном маршруте.

Заметим, что каждая вершина в G' имеет не более чем две входящие дуги и не более чем две исходящие. Задача коммивояжера на таком ориентированном графе эквивалентна задаче коммивояжера на кубическом ориентированном графе $G'' = (V'', \mathcal{A}'')$, где каждая вершина $v \in V'$ степени 4 заменена двумя вершинами $\check{v}, \hat{v}$, соединенными *фиктивной* дугой нулевой длины $(\check{v}, \hat{v})$, при этом в $\check{v}$ входят дуги, входившие в v, а из $\hat{v}$ выходят дуги, выходившие из v. Дуга $e \in \mathcal{A}''$ является предписанной и называется псевдодугой, если ей соответствует псевдодуга графа G'.

Решение последней задачи может быть получено перебором всех допустимых решений задачи коммивояжера с предписанными ребрами на вспомогательном неориентированном графе $\bar{G} = (V'', \bar{E})$, где пара вершин u, v соединяется ребром тогда и только тогда, когда эти вершины связывает дуга в графе G''. Ребро $\{u, v\} \in \bar{E}$ полагается предписанным, если (u, v) или (v, u) – псевдодуга или фиктивная дуга в графе G''. Множество предписанных ребер в $\bar{G}$ обозначим через $\bar{F}$. Выписать все гамильтоновы циклы в графе $\bar{G}$ с учетом предписанных ребер позволяет алгоритм трудоемкости $O(|V''| \cdot 2^{(|\bar{E}|-|\bar{F}|)/4})$, предложенный в [171]. Далее для любого гамильтонова цикла из $\bar{G}$ за время $O(|V''|)$ выясняется возможность провести по нему контуры в G'' в каждом из двух направлений и вычисляются длины таких контуров, если они существуют. Заметим, что $|\bar{E}| - |\bar{F}| = d \leq 2\hat{n}$, где d – число дуг, присутствующих ровно в одном из родительских маршрутов. Следовательно, трудоемкость решения задачи оптимальной рекомбинации на графе G равна $O(\hat{n} \cdot 2^{d/4})$ или $O(\hat{n} \cdot 1.42^{\hat{n}})$.

Стягивание путей, содержащихся в обоих родительских маршрутах, может существенно сократить трудоемкость алгоритма решения ЗОР, если имеется большое число совпадений между родительскими решениями.

Симметрический случай. Пусть в графе $G = (V, E)$ задано два родительских гамильтоновых цикла с множествами ребер E_1, E_2 и требуется решить задачу оптимальной рекомбинации. Построим сводимость данной задачи к задаче коммивояжера на графе со степенью вершин не более 4 и заданным подмножеством предписанных ребер. Подобно общему случаю, задача оптимальной рекомбинации сводится к задаче коммивояжера на графе $G' = (V', E')$, полученном из G исключением всех ребер, лежащих в $E \backslash (E_1 \cup E_2)$, и стягиванием цепей, содержащихся в обоих родительских циклах. Под стягиванием в данном случае понимается следующее отображение. Пусть некоторая цепь P_{uv} с концами в вершинах u и v является общим участком родительских циклов и не является частью какой-то другой цепи, содержащейся в обоих родительских циклах. Стягивание цепи P_{uv} переводит все ее вершины и ребра в одно предписанное ребро $\{u, v\}$ нулевой длины. Все прочие вершины и ребра графа остаются без изменений. Пусть F' обозначает множество ребер, предписанных в G' при стягивании всех цепей, содержащихся в обоих родительских циклах.

Степень вершин в G' не превышает четырех, а число вершин не превышает $\hat{n}$. Если оптимум в задаче коммивояжера с обязательным включением ребер из F' на графе G' некоторым образом уже получен, то замена предписанных ребер из F' на соответствующие им цепи в графе G обеспечивает выполнение условий ЗОР. Целевые функции двух этих задач отличаются на константу, равную общей длине исключенных цепей.

Для решения задачи коммивояжера на графе G' может быть использован рандомизированный алгоритм, предложенный в [171] для решения задачи коммивояжера на графах со степенью вершин не более 4 и заданным множеством предписанных ребер. Особенность алгоритма состоит в том, что кроме исходных данных задачи на вход подается значение p, указывающее требуемую вероятность получения оптимума. Если $p \in [0, 1)$ – константа, не зависящая от исходных данных задачи, то трудоемкость рандо-

мизированного алгоритма [171] составляет $O((27/4)^{\hat{n}/3})$, что есть $O(1.89^{\hat{n}})$.

Как отмечалось в п. 1.2.1, при использовании оператора кроссинговера в ГА указывается дополнительный параметр P_c, определяющий вероятность с которой выполняется рекомбинация особей. В таком случае естественно положить $p = P_c$, если $P_c \in [0, 1)$. Если же $P_c = 1$, то для оптимальной рекомбинации может быть использован дерандомизированный вариант алгоритма из [171], имеющий несколько большую трудоемкость.

3.3.6. Заключение

Полученные в настоящем разделе результаты показывают, что оптимальная рекомбинация для многих NP-трудных задач комбинаторной оптимизации также NP-трудна. Однако естественно ожидать, что зачастую ЗОР будет иметь существенно меньшую размерность по сравнению с исходной задачей, особенно после выполнения значительного числа итераций ГА, когда особи популяции приобретают много совпадающих генов. В такой ситуации даже NP-трудная ЗОР может успешно решаться современными методами ветвей и границ. Результаты использования операторов кроссинговера, в которых решаются NP-трудные ЗОР, приводятся в главе 5.

Снижение трудоемкости алгоритма оптимальной рекомбинации для задачи коммивояжера, предложенного в п. 3.3.5, может оказаться возможным за счет ускорения алгоритмов решения задачи коммивояжера на графе с ограниченными степенями вершин. В [43] указано представление решений, при котором «почти все» индивидуальные ЗОР для задачи о кратчайшем гамильтоновом пути полиномиально разрешимы. Рассмотренные в данном разделе ЗОР могут обладать аналогичными свойствами.

Кроме того представляет интерес экспериментальное исследование предложенных операторов оптимальной рекомбинации в составе генетических алгоритмов и их сравнение с операторами оптимальной рекомбинации, основанными на динамическом программировании [294].

4. Построение эволюционных алгоритмов со свойствами динамического программирования

В данной главе предложены многокритериальные эволюционные алгоритмы со свойствами, аналогичными свойствам методов динамического программирования. Показано, что в достаточно общих предположениях, при подходящем выборе операторов эволюционного алгоритма, его средняя трудоемкость полиномиально ограничена относительно трудоемкости алгоритма динамического программирования. Для класса задач, удовлетворяющих условиям существования вполне полиномиальной аппроксимационной схемы Г. Воегингера [293], предложена вполне полиномиальная рандомизированная аппроксимационная схема, основанная на эволюционном алгоритме.

В работах [35, 279, 286] теоретически показано, что некоторые известные задачи комбинаторной оптимизации могут быть решены эволюционными алгоритмами с приемлемой трудоемкостью, если в этих алгоритмах используется подходящий способ представления решений и соответствующий ему оператор мутации. Примеры данного подхода продемонстрированы на задаче о кратчайшем пути в графе [279], задаче коммивояжера [286] и задаче о рюкзаке [35]. Представление решений в этих работах существенно отличается от универсального способа, состоящего в использовании двоичной кодировки решений из формулировки задачи комбинаторной оптимизации. Вместо этого представление решений отражает некоторые свойства частичных решений задачи, что позволяет эволюционному алгоритму достраивать имеющиеся генотипы до оптимальных подобно тому, как это осуществляется в методе динамического программирования (ДП).

В настоящей главе предлагается обобщение упомянутых выше результатов в рамках единого подхода. Разработанный здесь ЭА соответствует схеме многокритериального ЭА (см. п. 1.2.2).

Ниже будет показано, что при достаточно широких предположениях задача, решаемая динамическим программированием за время T, может быть решена соответствующим эволюционным алгоритмом в среднем за время $O(T\bar{n}\log|DP|)$, где $\bar{n}$ – число стадий ДП, а $|DP|$ – число состояний, вычисленных за время работы ДП. При этом операторы ЭА имеют трудоемкость того же порядка, что и базовые операции ДП.

Результаты данной главы не дают более эффективных методов решения задач комбинаторной оптимизации, чем известные ранее. Основной целью является уточнение класса задач, которые могут быть успешно решены с помощью ЭА.

Глава организована следующим образом. В разделе 4.1 дается общая схема алгоритма динамического программирования и описывается вид рассматриваемых задач. Данная схема динамического программирования преобразуется в эволюционный алгоритм в разделе 4.2. Там же приводится ряд примеров построения ЭА на основе динамического программирования для некоторых известных задач. В разделе 4.3 для класса задач комбинаторной оптимизации, удовлетворяющих условиям существования вполне полиномиальной аппроксимационной схемы Г. Воегингера [293], предложена вполне полиномиальная рандомизированная аппроксимационная схема, основанная на эволюционном алгоритме.

4.1. Формализация метода динамического программирования

Применение метода динамического программирования ко многим задачам комбинаторной оптимизации может рассматриваться как сведение исходной задачи П к некоторой вспомогательной задаче многокритериальной оптимизации П′, при этом последняя решается с помощью

«стандартного» алгоритма динамического программирования. Для простоты здесь будем предполагать непустоту множества допустимых решений, т. е. $\mathrm{Sol}(I) \neq \emptyset$.

4.1.1. Вспомогательная задача многокритериальной оптимизации

Рассмотрим вспомогательную задачу многокритериальной оптимизации Π', при этом в дополнение к постановке задачи многокритериальной оптимизации из п. 1.2.2 введем множество допустимых решений задачи и формализуем понятие индивидуальной задачи.

Пусть имеется $d' \in \mathbb{N}$ критериев оптимизации (максимизации или минимизации). Индивидуальная задача I' из многокритериальной задачи Π' определяется четверкой $(d', g, \mathcal{S}, \mathcal{D})$. Здесь $\mathcal{S}$ – *пространство поиска*, $g\colon \mathcal{S} \to (\mathbb{R} \cup \{\infty\})^{d'}$ – векторнозначная целевая функция, $g(\mathcal{S}) \subseteq \mathbb{R}^{d'}$ – *пространство критериев* и $\mathcal{D} \subseteq \mathcal{S}$ – множество допустимых решений.

Введем частичный порядок $\succeq$ для обозначения доминирования по Парето: $(y'_1, \ldots, y'_{d'}) \succeq (y_1, \ldots, y_{d'})$ тогда и только тогда, когда $y'_j \leq y_j$ для всех j, таких что g_j – критерий минимизации, и $y'_j \geq y_j$ для всех j таких что g_j – критерий максимизации.

В дальнейшем будем использовать обозначение $\mathbf{y}' \succ \mathbf{y}$ в случае, когда одновременно выполняется $\mathbf{y}' \succeq \mathbf{y}$ и $\mathbf{y} \not\succeq \mathbf{y}'$. *Фронтом Парето* будем называть множество максимальных элементов по отношению $\succeq$ во множестве $g(\mathcal{D})$. Решение задачи состоит в отыскании полного множества альтернатив (ПМА) $\mathcal{S}^*$, т. е. минимального по включению множества в $\mathcal{D}$, которое отображается во фронт Парето под действием g.

Предположим, что построение индивидуальной задачи I' по индивидуальной задаче I осуществляется за полиномиальное время. ПМА для задачи Π' некоторой эффективной процедурой преобразуется в оптимальное решение $\mathbf{x}^* \in \mathrm{Sol}(I)$ индивидуальной задачи I из Π. Данная процедура, как

правило, состоит в применении метода обратного хода (см., например, [70], гл. 15), который может быть ускорен, если в процессе работы ДП сохраняется дополнительная информация о том, каким образом вычисляется каждое новое состояние.

4.1.2. Общая схема динамического программирования

Алгортим ДП для задачи Π' выполняется за некоторое число итераций, называемых далее *стадиями*. На каждой стадии ДП вычисляется и записывается в память некоторое множество *состояний*, принадлежащих пространству $\mathcal{S}$. Далее будем рассматривать только такие алгоритмы ДП, где всякое новое состояние текущей стадии вычисляется с помощью некоторой *переходной функции*, причем каждая такая функция имеет своим аргументом одно из состояний, вычисленных на предыдущей стадии, и, вообще говоря, зависит от исходных данных индивидуальной задачи I' из многокритериальной задачи Π'.

Перейдем к более детальному описанию ДП. Для этого рассмотрим сначала *упрощенный* алгоритм ДП. Пусть данный алгоритм выполняется за $\bar{n}$ стадий и на стадии i вычисляется множество состояний $\mathcal{S}_i \subseteq \mathcal{S}$. Для описания ДП потребуется $\bar{n}$ конечных множеств отображений $\mathcal{F}_i$, $i = 1, \ldots, \bar{n}$, состоящих из переходных функций вида $F: \mathcal{S} \to \mathcal{S}'$. Здесь $\mathcal{S}'$ является расширением пространства $\mathcal{S}$ или совпадает с ним. Результатом действия отображения F может оказаться состояние $F(S) \in \mathcal{S}' \backslash \mathcal{S}$, не принадлежащее пространству поиска. С целью исключения таких элементов на каждой стадии i, $i = 1, \ldots, \bar{n}$, используется *функционал допустимости* H_F, $H_F: \mathcal{S} \to \mathbb{R}$, такой что при любом $F \in \mathcal{F}_i$ выполнено $F(S) \in \mathcal{S}$ тогда и только тогда, когда $H_F(S) \leq 0$. Далее полагаем, что число $\bar{n}$, все функции H_F и множества функций $\mathcal{F}_i$ зависят от входных данных индивидуальной задачи из многокритериальной задачи Π'.

Упрощенный алгоритм ДП выполняется следующим образом. На эта-

пе инициализации формируется множество $\mathcal{S}_0$, являющееся конечным подмножеством $\mathcal{S}$. На i-той стадии вычисляется множество состояний $\mathcal{S}_i$ с использованием уже имеющегося множества $\mathcal{S}_{i-1}$ по формуле

$$\mathcal{S}_i = \{F(S) \mid S \in \mathcal{S}_{i-1},\ F \in \mathcal{F}_i,\ H_F(S) \leq 0\}. \tag{4.1}$$

Заметим, что после завершения стадии $\bar{n}$ множество $\mathcal{S}_{\bar{n}}$ может содержать такие состояния, в которых значения целевой функции доминируются по Парето значениями целевой функции некоторых других состояний из $\mathcal{S}_{\bar{n}}$. В связи с этим в качестве результата упрощенного алгоритма ДП вычисляется минимальное по включению подмножество $\mathcal{S}'_{\bar{n}} \subseteq \mathcal{S}_{\bar{n}}$, такое что $g(\mathcal{S}'_{\bar{n}}) = g(\mathcal{S}_{\bar{n}})$.

Для исключения состояний, значения целевой функции которых доминируются по Парето, а также для снижения трудоемкости в приложениях ДП, как правило, используется принцип оптимальности Р. Беллмана (см., например, [130]) или такие его обобщения, как в алгоритме последовательного анализа вариантов В.С. Михалевича и Н.З. Шора [79] (см. алгоритм «киевский веник» в [80]). Это позволяет удалить из рассмотрения неперспективные состояния, не нарушая оптимальности результата. Достаточные условия применимости принципа Р. Беллмана в случае задач с одним критерием оптимизации сформулированы в [241]. Во многокритериальном случае классический принцип Беллмана, как правило, неприменим, однако неперспективные состояния могут быть удалены с помощью соответствующего отношения доминирования на множестве состояний ДП – см., например, [61, 79, 228]. В настоящей главе используется аналогичный подход, согласованный с системой обозначений из [293].

Рассмотрим предпорядок $\preceq_{\text{dom}}$ (т. е. рефлексивное и транзитивное отношение) на множестве $\mathcal{S}'$, такой что $S \preceq_{\text{dom}} S'$ тогда и только тогда, когда 1) $g(S) \preceq g(S')$ или 2) $S \notin \mathcal{S},\ S' \in \mathcal{S}$. Будем называть состояние S *доминируемым* состоянием S', если $S \preceq_{\text{dom}} S'$. Если $S \in \mathcal{T} \subseteq \mathcal{S}$, такое что не существует $S' \in \mathcal{T}$, $S \preceq_{\text{dom}} S'$, то S будем называть *недоминируемым* в $\mathcal{T}$.

Как будет показано в дальнейшем, два следующих условия позволяют использовать отношение $\preceq_{\text{dom}}$ для исключения неперспективных состояний. Первое условие дает гарантию того, что отношение доминирования между парой состояний переносится и на их образы при действии переходных функций.

Условие С.1. При любых $S, S' \in \mathcal{S},\ i = 1, \ldots, \bar{n}$, если $S \preceq_{\text{dom}} S'$, то $F(S) \preceq_{\text{dom}} F(S')$ для всех $F \in \mathcal{F}_i$ таких, что $F(S') \in \mathcal{S}$.

Второе условие означает, что при действии переходных функций образ доминирующего состояния имеет не большее значение функционала допустимости, чем образ доминируемого.

Условие С.2. Для любых $S, S' \in \mathcal{S},\ i = 1, \ldots, \bar{n}$ и $F \in \mathcal{F}_i$, если $S \preceq_{\text{dom}} S'$, то $H_F(S') \leq H_F(S)$.

Рассмотрим подмножество $\mathcal{S}_i$ пространства состояний $\mathcal{S}$. Будем называть $\mathcal{T}_i \subseteq \mathcal{S}_i$ *доминирующим подмножеством* множества $\mathcal{S}_i$, если для любого состояния $S \in \mathcal{S}_i$ существует $S' \in \mathcal{T}_i$, такое что $S \preceq_{\text{dom}} S'$. Обозначим через $M(\mathcal{S}_i)$ множество минимальных по включению доминирующих подмножеств в $\mathcal{S}_i$.

Следующее утверждение показывает, что при условиях С.1 и С.2 в алгоритме ДП достаточно хранить не все множество $\mathcal{S}_i$, а лишь его доминирующее подмножество.

Утверждение 4.1. *Пусть упрощенный алгоритм ДП вычисляется согласно (4.1), условия С.1 и С.2 выполнены,* $\mathcal{T}_0 \in M(\mathcal{S}_0)$ *и доминирующие множества* $\mathcal{T}_i,\ i = 1, \ldots, \bar{n}$, *вычисляются таким образом, что*

$$\mathcal{T}_i \in M(\{F(S) \mid S \in \mathcal{T}_{i-1} \wedge F \in \mathcal{F}_i \wedge H_F(S) \leq 0\}). \tag{4.2}$$

Тогда для любого состояния $S^* \in \mathcal{S}_i,\ i = 0, \ldots, \bar{n}$, *найдется* $S \in \mathcal{T}_i$, *такое что* $S^* \preceq_{\mathrm{dom}} S$.

Доказательство. Проведем индукцию по i. Для $i = 0$ утверждение верно по предположению $\mathcal{T}_0 \in M(\mathcal{S}_0)$.

Согласно (4.1) состояние $S^* \in \mathcal{S}_i$ может быть выражено как $S^* = F^*(S')$, причем $H_{F^*}(S') \leq 0$, $F^* \in \mathcal{F}_i$, и $S' \in \mathcal{S}_{i-1}$. Однако по предположению индукции найдется состояние $S^\diamond \in \mathcal{T}_{i-1}$, такое что $S' \preceq_{\mathrm{dom}} S^\diamond$. Тогда из условий С.1 и С.2 вытекает, что $S^* = F^*(S') \preceq_{\mathrm{dom}} F^*(S^\diamond)$ и $F^*(S^\diamond) \in \{F(S) \mid S \in \mathcal{T}_{i-1},\ F \in \mathcal{F}_i,\ H_F(S) \leq 0\}$. Следовательно, ввиду (4.2) существует $S \in \mathcal{T}_i$, такое что $S^* \preceq_{\mathrm{dom}} F^*(S^\diamond) \preceq_{\mathrm{dom}} S$. □

По определению предпорядка $\preceq_{\mathrm{dom}}$ если условия утверждения 4.1 выполняются и упрощенный алгоритм ДП вычисляет ПМА для задачи I', то множество $\mathcal{T}_{\bar{n}}$ является ПМА для I'.

Для простоты дальнейших рассуждений предположим, что $M(\mathcal{S}_0) = \{\mathcal{S}_0\}$, т. е. ни один элемент множества $\mathcal{S}_0$ не доминируется другим его элементом. Данное предположение в известных алгоритмах ДП, как правило, выполняется. Для большей общности это предположение, однако, может быть снято (см., например, [156]).

Позднее при доказательстве леммы 4.1 будет использовано следующее

Утверждение 4.2. *Если условия утверждения 4.1 выполняются, то мощность каждого множества* $\mathcal{T}_i,\ i = 0, \ldots, \bar{n}$, *определена однозначно.*

В самом деле рассмотрим множество максимальных элементов множества $\mathcal{S}_i$ по отношению $\preceq_{\mathrm{dom}}$. Определим классы эквивалентности в этом множестве на основе отношения эквивалентности

$$S \approx S' \Leftrightarrow (S \preceq_{\mathrm{dom}} S' \text{ и } S' \preceq_{\mathrm{dom}} S)\,.$$

Мощность минимального подмножества $M_{\max}$ максимальных элементов, такого что $M_{\max}$ доминирует все множество $\mathcal{S}_i$, определяется однозначно,

так как $M_{\max}$ неизбежно содержит по одному представителю из каждого класса эквивалентности. $\square$

Процесс вычислений согласно выражению (4.2) может быть представлен в алгоритмической форме:

Алгоритм 4.1. Динамическое программирование для задачи Π'

1. $\mathcal{T}_0 := \mathcal{S}_0$.
2. Для всех $i = 1 \ldots \bar{n}$:
3. $\quad \mathcal{T}_i := \emptyset$.
4. $\quad$ Для всех $S \in \mathcal{T}_{i-1}$ & $F \in \mathcal{F}_i$:
5. $\quad\quad$ Если $H_F(S) \leq 0$ & $\nexists S' \in \mathcal{T}_i$: $F(S) \preceq_{\text{dom}} S'$, то
6. $\quad\quad\quad \mathcal{T}_i := (\mathcal{T}_i \setminus \{S' \in \mathcal{T}_i \mid S' \prec_{\text{dom}} F(S)\}) \cup \{F(S)\}$.
7. $\quad$ Конец цикла.
8. Конец цикла.
9. Результат: $\mathcal{T}_{\bar{n}}$.

Легко видеть, что, когда вычислено подмножество $\mathcal{T}_i$ по окончании цикла из строк 4-8, имеет место соотношение (4.2). Таким образом, если упрощенный алгоритм ДП решает задачу Π' и условия С.1 и С.2 выполнены, то по утверждению 4.1 алгоритм 4.1 также решает Π'.

Изложенная общая схема ДП отличается от общих схем многокритериального ДП из [61, 228] главным образом тем, что в [61, 228] используется иное понятие состояния, включающее в одно состояние целое семейство однотипных состояний в терминах рассматриваемой здесь схемы ДП. Однотипность этих состояний выражается в [228] условием *сепарабельности*. Условиям вида С.1 и С.2 в общей схеме ДП [228] в некоторой мере соответствует условие *монотонности*. В [61] используется весьма ограничительное предположение об аддитивности критериев, но отсутствует явное

разделение вычислительного процесса на стадии и не требуется выполнения условий вида C.1 и C.2.

С целью упрощения дальнейшего анализа будем считать, что в строке 5 проверка условия $\nexists S' \in \mathcal{T}_i \colon F(S) \preceq_{\text{dom}} S'$ выполняется всякий раз независимо от знака $H_F(S)$.

Время выполнения алгоритма ДП зависит от трудоемкости вычисления переходных функций $F \in \mathcal{F}_i$, функционалов допустимости H_F, проверки отношения доминирования и действий с множествами состояний. Под *обработкой состояния* в ДП будем понимать итерацию внутреннего цикла, т. е. строки 5 и 6 алгоритма 4.1. За время работы алгоритма ДП обработка состояния выполняется $t_{\text{ДП}}$ раз, где $t_{\text{ДП}} = \sum_{i=0}^{\bar{n}-1} |\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|$.

Обозначим через τ и τ' соответственно верхнюю и нижнюю оценки трудоемкости обработки состояния. В таком случае общее время вычислений ДП содержится в интервале $[t_{\text{ДП}}\tau', t_{\text{ДП}}\tau]$.

Заметим, что время вычислений ДП полиномиально ограничено относительно длины входных данных индивидуальной задачи I', если $\bar{n}$, τ так же, как $|\mathcal{T}_i|$ и $|\mathcal{F}_{i+1}|$ при $i = 0, \ldots, \bar{n} - 1$, полиномиально ограничены относительно длины входа I'.

Во многих приложениях ДП вычисление переходных функций и функционалов допустимости выполняются за константное время. Кроме того, предпорядок $\succeq_{\text{dom}}$ на $\mathcal{S}$ зачастую представляет собой произведение линейных порядков и на каждой стадии достаточно хранить всего один элемент (максимальный среди найденных) для каждого линейного порядка. Такая структура данных позволяет выполнять обработку состояния за время $\Theta(1)$, и имеет место

Утверждение 4.3. *Пусть время обработки состояния есть* $\Theta(1)$. *Тогда общее время счета ДП составляет*

$$\Theta\left(\sum_{i=0}^{\bar{n}-1} |\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|\right). \tag{4.3}$$

4.1.3. Примеры применения общей схемы динамического программирования

Рассмотрим несколько примеров применения представленной выше общей схемы ДП к некоторым классическим задачам комбинаторной оптимизации. Во всех этих примерах $\preceq_{\mathrm{dom}}$ на $\mathcal{S}$ является произведением линейных порядков. Далее символ id обозначает переходную функцию, действующую как тождественное отображение.

Задача коммивояжера. Применим описанную выше общую схему ДП к задаче коммивояжера (см. постановку задачи в разделе 1.1). Пусть $\bar{n} = |V|$ – число вершин графа.

Пространство состояний $\mathcal{S}$ задачи Π', соответствующей алгоритму динамического программирования М. Хельда и Р. Карпа [211], состоит из простых путей $S = (v_{k_1}, \ldots, v_{k_i})$, где $k_1 = 1$, имеющих чило вершин $i = 1, \ldots, \bar{n}$. Расширенное пространство состояний $\mathcal{S}'$ состоит из всевозможных путей длины не более $\bar{n}$, начинающихся в вершине v_1 (одна вершина при этом может входить в последовательность $S \in \mathcal{S}'$ более одного раза). При $M \subseteq V\backslash\{v_1\}$ и $v_k \in M$ обозначим через $\pi(k, M)$ множество простых путей, состоящих из $|M| + 1$ вершины, начинающихся в вершине v_1, затем проходящих по всему множеству M и оканчивающихся в v_k. Пусть векторнозначная целевая функция $g : \mathcal{S} \to (\mathbb{R} \cup \{\infty\})^{d'}$, $d' = (\bar{n} - 1)2^{(\bar{n}-1)}$ имеет значения $g_{kM}(S)$, $M \subseteq V\backslash\{v_1\}$, $v_k \in M$, равные длине пути S в том и только том случае, когда $S \in \pi(k, M)$. Для всех других $S \notin \pi(k, M)$ положим $g_{kM}(S) = \infty$. Множество допустимых решений $\mathcal{D} = \cup_{k=2}^{\bar{n}} \pi(k, V\backslash\{v_1\})$.

Начальное множество $\mathcal{S}_0$ состоит из одного элемента v_1. Множество $\mathcal{F}_i$ для всякого i состоит из $\bar{n} - 1$ функции $F_v{:}\,\mathcal{S} \to \mathcal{S}'$, каждая из которых добавляет соответствующую вершину $v \in V\backslash\{v_1\}$ в конец простого пути, заданного в качестве аргумента. В случае если состояние $S' = F_v(S)$ недопустимо, т. е. S' не является простым путем, функционал $H_F(S)$ имеет

значение 1. На прочих аргументах его значение равно 0.

С учетом выбора функции g отношение доминирования определяется следующим образом. $S \preceq_{\mathrm{dom}} S'$ тогда и только тогда, когда: 1) либо S и S' – простые пути, проходящие по одному множеству вершин и заканчивающиеся в v_k, причем путь S' не длиннее, чем S; 2) либо S не является простым путем, а S' является. Состояния из разных множеств $\pi(k, M)$ не сравнимы между собой. Условия С.1 и С.2 проверяются непосредственно.

Описанный выше общий алгоритм ДП с выбранными компонентами для ЗК составляет основную часть алгоритма М. Хельда и Р. Карпа [211] кроме его последнего шага, на котором строится оптимальное решение ЗК на основе семейства кратчайших гамильтоновых путей. Данный шаг соответствует вычислению оптимального решения индивидуальной задачи I из задачи П на основе ПМА задачи I'. Для этого достаточно выбрать из $\mathcal{T}_{\bar{n}}$ состояние, минимизирующее величину $g_{k,V\setminus\{v_1\}}(S) + c_{k1},\ k \neq 1$. Использование подходящих структур данных позволяет обеспечить константные трудоемкости вычисления переходных функций, функционалов H_F и обработки состояния в целом. При этом $|\mathcal{T}_i| = i\binom{\bar{n}-1}{i}$, $|\mathcal{F}_i| = \bar{n} - 1$ для всех $i = 1, \ldots, \bar{n}$. В таком случае из утверждения 4.3 вытекает, что общая трудоемкость ДП есть $\Theta(|V|^2 2^{|V|})$.

Задача об одномерном булевом рюкзаке. Другим известным примером задач комбинаторной оптимизации, разрешимых с помощью ДП, является задача об одномерном булевом рюкзаке (см. раздел 1.1). Рассмотрим эту задачу в рамках общей схемы ДП, описанной выше. В данном случае $\bar{n} = n$. Пусть каждое состояние $S = (s_1, s_2) \in \mathcal{S}_i,\ i = 1, \ldots, n$, соответствует частичному решению, относящемуся к первым i предметам, где компонента s_1 содержит вес предметов, выбранных в частичном решении, а компонента s_2 – их суммарную полезность. Начальное множество $\mathcal{S}_0$ состоит из одного элемента $(0, 0)$, соответствующего решению, в котором не выбран

ни один предмет.

Векторный критерий минимизации $g\colon \mathcal{S} \to \mathbb{R}^A$ имеет вид

$$g_a(S) := \begin{cases} s_2 & \text{если } s_1 = a \\ 0 & \text{иначе} \end{cases}, \quad a = 0, \ldots, A. \tag{4.4}$$

Пусть $S = (s_1, s_2),\ S' = (s'_1, s'_2)$. В таком случае $S \preceq_{\text{dom}} S'$ тогда и только тогда, когда 1) либо $s_1 = s'_1$ и $s_2 \le s'_2$; 2) либо $s_1 > A$ и $s'_1 \le A$.

Множество $\mathcal{F}_i$ состоит из двух функций: id и $F_i(s_1, s_2) = (s_1 + a_i, s_2 + c_i)$. Здесь F_i соответствует включению в частичное решение предмета i, а функция id соответствует пропуску этого элемента. Новое состояние $S = (s_1, s_2)$ принимается к рассмотрению, если оно не нарушает ограничение на общий вес предметов, т. е. $H_{F_i}(S) \le 0$, где $H_{F_i}(S) = s_1 + a_i - A$.

Условия С.1 и С.2 проверяются непосредственно. Для получения оптимального решения задачи достаточно выбрать состояние в $\mathcal{T}_n$, имеющее наибольшее значение компоненты s_2.

С целью сокращения трудоемкости процедуры сравнения имеет смысл организовать хранение состояний ДП в матрице размерности $(A \times n)$. Элемент в строке a, $a = 1, \ldots, A$, и столбце i, $i = 1, \ldots, n$, содержит лучшее найденное значение s_1 для всех вычисленных состояний вида $S = (s_1, s_2) \in \mathcal{T}_i$, где $s_2 = a$. Тогда, полагая τ равной константе, получаем трудоемкость ДП $O(nA)$, так как $\sum_{i=1}^{n} |\mathcal{T}_{i-1}| \le nA$.

Задача о кратчайшем пути между двумя вершинами. Рассмотрим задачу о кратчайшем пути между двумя вершинами: дан связный граф $G = (V, E)$, где $V = \{v_1, \ldots, v_{|V|}\}$ – множество вершин, E – множество ребер, и заданы положительные веса его ребер $w\colon E \to \mathbb{R}^+$. Задача состоит в отыскании кратчайшего пути из вершины *источника* $v_{\text{s}} = v_1$ в *целевую* вершину $v_{\text{t}} = v_{|V|}$.

Пусть $\bar{n} = |V|$. Пространство состояний $\mathcal{S}$ представляет собой множество простых путей в графе G, начинающихся в вершине v_{s}. Множество допустимых решений $\mathcal{D}$ совпадает с $\mathcal{S}$.

Ввиду того, что добавление вершины к простому пути может привести к повтору имеющейся в пути вершины, рассмотрим расширенное пространство поиска $\mathcal{S}'$, состоящее из всевозможных путей длиной не более $\bar{n}$, начинающихся с вершины v_{s}. Множество $\mathcal{S}_0$ начальных решений состоит из одного элемента v_{s}. Для всех i, $i = 1, \ldots, \bar{n}$, определим $\mathcal{F}_i := \{F_v \mid v \in V\} \cup \{\mathrm{id}\}$, где $F_v \colon \mathcal{S} \to \mathcal{S}'$ – отображение, состоящее в добавлении вершины v к указанной последовательности вершин. Положим $H_{F_v}(S) = -1$, если $F_v(S)$ – простой путь в G, начинающийся в вершине v_{s}; в противном случае $H_{F_v}(S) = 1$.

Пусть векторная целевая функция g имеет $d' = \bar{n}$ компонент $g_v(S)$, $v \in V$, значение каждой из которых равно длине пути S, если только S – простой путь, проходящий от вершины v_{s} к вершине v; в противном случае полагаем $g_v(S) = \infty$. Таким образом, $S \prec_{\mathrm{dom}} S'$ в том и только том случае, когда: 1) либо простые пути S и S' ведут из v_{s} в одну и ту же вершину и путь S' короче пути S; 2) либо S не является простым путем, а S' является.

Общая схема ДП применима к данной задаче, $\tau' = \Omega(1)$, $\tau = O(1)$, и $|\mathcal{T}_{i-1}| = \Theta(\bar{n})$, $|\mathcal{F}_i| = \Theta(\bar{n})$ для всех $i = 1, \ldots, \bar{n}$. Для получения кратчайшего пути между вершинами v_{s} и v_{t} достаточно выбрать в $\mathcal{T}_{\bar{n}}$ состояние S с наименьшим значением $g_{v_{\mathrm{t}}}(S)$. Ввиду (4.3) общая трудоемкость алгоритма 4.1 составляет $\Theta(|V|^3)$. С той же трудоемкостью может быть найдено множество всех кратчайших путей от вершины v_{s} – в данном случае это все ПМА задачи I'.

Заметим, что известный алгоритм Дейкстры имеет оценку трудоемкости $O(|V|^2)$, но в отличие от рассматриваемой общей схемы ДП в алгоритме Дейкстры на каждой стадии переходная функция применяется однократно (при этом ближайшая вершина присоединяется к множеству вершин,

достигнутых ранее). Такие специальные модификации ДП в рамках общей схемы не рассматриваются.

Применение общей схемы ДП к задаче поиска кратчайших путей между всеми парами вершин графа осуществляется аналогичным образом [156].

4.2. Эволюционные алгоритмы на основе динамического программирования

Рассмотрим возможности эволюционных алгоритмов в тех случаях, когда к задаче применима общая схема ДП, описанная выше. Для этого сначала изложим подобную ей общую схему ЭА для задачи Π', базирующегося на ДП, затем оценим трудоемкость такого ЭА и рассмотрим примеры применения этой общей схемы.

4.2.1. Общая схема эволюционного алгоритма на основе динамического программирования

Пусть $\mathcal{B} := \{0, \ldots, \bar{n}\} \times \mathcal{S}'$ – множество генотипов ЭА, где первая компонента генотипа указывает номер стадии, к которой относится состояние, закодированное во второй компоненте (конкретный способ кодировки обеих компонент не имеет значения). Множеством фенотипов является $\mathcal{S}'$. Функция $\Psi : \mathcal{B} \to \mathcal{S}'$ действует как проекция генотипа $\xi = (i, S) \in \mathcal{B}$ в точку $S \in \mathcal{S}'$. В предлагаемом ЭА естественным образом определяется многокритериальная функция приспособленности $\Phi(\xi) = (\Phi_1(\xi), \ldots, \Phi_{d'}(\xi))$ на основе d'-мерной целевой функции g задачи I'. Если $\xi = (i, S)$, $i \in \{0, \ldots, \bar{n}\}$, $S \in \mathcal{S}$, то для всех j, таких что g_j – критерий максимизации, полагаем $\Phi_j(\xi) = g_j(S)$; для всех остальных j полагаем $\Phi_j(\xi) = -g_j(S)$. Если же $S \in \mathcal{S}' \backslash \mathcal{S}$, то $\Phi_j(\xi) = -\infty$, $j = 1, \ldots, d'$.

Заметим, что на основе векторнозначной целевой функции g посред-

ством $\preceq_{\text{dom}}$ можно определить предпорядок $\preceq_{\text{EA}}$ на множестве генотипов:

$$(i, S) \preceq_{\text{EA}} (i', S') \Leftrightarrow (i = i' \text{ и } S \preceq_{\text{dom}} S'). \tag{4.5}$$

Использование данного предпорядка вместо векторной функции приспособленности в дальнейшем позволит упростить описание предлагаемого многокритериального ЭА.

В процессе работы ЭА для построения новых особей используются оператор селекции $\text{Sel}^{\diamond}$ и оператор мутации $\text{Mut}^{\diamond}$, которые будут описаны ниже. ЭА для задачи Π' выглядит следующим образом.

Алгоритм 4.2. Эволюционный алгоритм для задачи Π'

1. $\hat{X} := \{0\} \times \mathcal{S}_0$.
2. Пока не выполнен критерий остановки :
3. $\quad \xi := \text{Mut}^{\diamond}(\text{Sel}^{\diamond}(\hat{X}))$.
4. $\quad$ Если $\nexists \xi' \in \hat{X}$, такой что $\xi \preceq_{\text{EA}} \xi'$, то
5. $\quad\quad \hat{X} := (\hat{X} \setminus \{\xi' \in \hat{X} \mid \xi' \prec_{\text{EA}} \xi\}) \cup \{\xi\}$.
6. $\quad$ Конец если.
7. Конец цикла.
8. Результат: $\{\Psi(\xi) \mid \xi = (i, S) \in \hat{X},\ S \in \mathcal{D}\}$.

Оператором селекции $\text{Sel}^{\diamond}$ выбирается особь $\xi \in \hat{X}$. Для этого, во-первых, выбирается индекс $i \leq \bar{n} - 1$ равновероятно на множестве стадий, представленных в особях текущей популяции, т. е. из $\{k : k \leq \bar{n} - 1,\ \exists (k, S) \in \hat{X}\}$. Во-вторых, равновероятно (с возвращением) выбирается генотип ξ среди особей вида (i, S) в текущей популяции X.

При заданном входном генотипе $\xi = (i, S)$ в операторе мутации $\text{Mut}^{\diamond}$ равновероятно выбирается переходная функция $F \in \mathcal{F}_{i+1}$ и при условии $H_F(S) \leq 0$ результат мутации имеет вид $\text{Mut}^{\diamond}((i, S)) = (i + 1, F(S))$; иначе (при $H_F(S) > 0$) $\text{Mut}^{\diamond}(\xi) = \xi$. Вопрос о соответствии оператора $\text{Mut}^{\diamond}$ известным биологическим механизмам мутации обсуждается в [156].

На последнем шаге ЭА применяется функция $\Psi((i,S)) = S$ для исключения дополнительной информации о номере стадии с целью получить набор допустимых решений задачи I'. По умолчанию будем предполагать, что в алгоритме 4.2 критерий остановки никогда не выполняется.

Предложенный алгоритм соответствует общей схеме многокритериального ЭА [257], где численность промежуточной популяции $N = 1$ (см. алгоритм 1.4).

4.2.2. Средняя трудоемкость эволюционного алгоритма

Обозначим наибольшую из мощностей множеств состояний, вычисленных за все время работы ДП, через $W_{\max}$, т. е. $W_{\max} = \max_{i=0,\ldots,\bar{n}} |\mathcal{T}_i|$. Заметим, что величина $W_{\max}$ определена корректно, так как мощности множеств $\mathcal{T}_i$ определяются однозначно, как следует из утверждения 4.2.

Лемма 4.1. *Если задача* Π' *разрешима алгоритмом ДП и условия С.1, С.2 выполнены, то математическое ожидание числа итераций эволюционного алгоритма 4.2 до построения ПМА есть*

$$O\left(\bar{n} \cdot \log W_{\max} \cdot \sum_{i=0}^{\bar{n}-1} |\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|\right).$$

Доказательство. Воспользуемся рассуждениями, аналогичными доказательствам теоремы 7 в [279] и теоремы 1 в [35]. Рассмотрим работу ЭА как поэтапный процесс, где этап $i,\ i = 0,\ldots,\bar{n}$, завершается, когда для каждого состояния $S \in \mathcal{T}_i$ построена особь $(i,S') \in \hat{X}$, такая что S' доминирует S. Здесь и далее в настоящем доказательстве через $\mathcal{T}_i,\ i = 0,\ldots,\bar{n}$, обозначаются те же подмножества, что вычисляются в алгоритме 4.1. Заметим, что после завершения этапа i подмножество генотипов, имеющих вид (i,S) в популяции X, уже не будет изменяться на последующих итерациях ЭА. Пусть $\mathcal{T}'_i$ обозначает множество состояний, представленных в таких особях после завершения этапа $i,\ i = 0,\ldots,\bar{n}$. По схеме алгорит-

ма 4.2 последовательность $\mathcal{T}'_i,\ i = 0, \ldots, \bar{n}$, удовлетворяет условию (4.2), и, следовательно, $|\mathcal{T}'_i| = |\mathcal{T}_i|,\ i = 0, \ldots, \bar{n}$, ввиду утверждения 4.2.

Пусть случайная величина θ_i равна числу итераций с момента окончания этапа $i-1$ до окончания этапа $i,\ i = 1, \ldots, \bar{n}$. Тогда математическое ожидание числа итераций ЭА до получения ПМА равно $\mathbf{E}[\theta]$, где $\theta = \theta_1 + \ldots + \theta_{\bar{n}}$.

Любое состояние $S \in \mathcal{T}_{i+1}$ вычисляется в алгоритме 4.1 посредством некоторой функции $\tilde{F} \in \mathcal{F}_{i+1}$, примененной к одному из состояний $\tilde{S} \in \mathcal{T}_i$. Таким образом, на этапе $i+1$ во время мутации та же переходная функция $\tilde{F}$ может быть применена к некоторой особи $\xi' = (i, S')$, такой что $\tilde{S} \preceq_{\text{dom}} S'$. После этой мутации, ввиду условий С.1 и С.2, популяция X будет содержать особь $\xi'' = (i+1, S'')$, где S'' – состояние, такое что $S \preceq_{\text{dom}} \tilde{F}(S') \preceq_{\text{dom}} S''$.

Рассмотрим произвольную итерацию ЭА на этапе $i+1$. Пусть k – число состояний из $\mathcal{T}_{i+1}$, которые уже доминируются какими-то состояниями, представленными в особях популяции X. Тогда найдется $|\mathcal{T}_{i+1}| - k$ новых генотипов вида $(i+1, S)$, которые могут быть добавлены в популяцию X, таких что после их добавления этап $i+1$ будет окончен (напомним, что $|\mathcal{T}'_{i+1}| = |\mathcal{T}_{i+1}|$). Вероятность получения особи (i, S'), где S' доминирует какое-либо из ранее не доминированных состояний множества $\mathcal{T}_{i+1}$, составляет не менее $(|\mathcal{T}_{i+1}| - k)/(\bar{n}|\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|)$. По свойствам геометрического распределения среднее число испытаний до наступления первого такого события не превышает $(\bar{n}|\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|)/(|\mathcal{T}_{i+1}| - k)$. Это означает, что математическое ожидание числа итераций на этапе $i+1$ ограничивается сверху:

$$\mathbf{E}[\theta_{i+1}] \leq \sum_{t'=1}^{|\mathcal{T}_{i+1}|} \frac{\bar{n}|\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|}{t'} = \bar{n}|\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}| \cdot \mathcal{H}(|\mathcal{T}_{i+1}|),$$

где $\mathcal{H}(K)$ обозначает K-е гармоническое число, $\mathcal{H}(K) = \sum_{j=1}^{K} 1/j$. Следо-

вательно,

$$\mathbf{E}[\theta] \leq \sum_{i=0}^{\bar{n}-1} \bar{n}|\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}| \cdot \mathcal{H}(|\mathcal{T}_{i+1}|) \leq \bar{n}(\ln W_{\max} + 1) \cdot \sum_{i=0}^{\bar{n}-1} |\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|.$$

□

Если в алгоритме 4.2 популяция хранится в памяти как набор непересекающихся подмножеств генотипов в соответствии со значениями компоненты i, $i = 0, \ldots, \bar{n}$, то средняя трудоемкость предложенного ЭА есть

$$O\left(\bar{n} \log W_{\max} \cdot \sum_{i=0}^{\bar{n}-1} (|\mathcal{F}_{i+1}| \cdot |\mathcal{T}_i| \cdot \tau)\right).$$

Сравнение ЭА и ДП предполагает отыскание для алгоритма ДП некоторого ЭА и анализ трудоемкости этих двух алгоритмов. Однако общая схема многокритериального ЭА, приведенная в разделе 1.2.2, имеет достаточно общий вид, допускающий, например, внедрение исходного алгоритма ДП в любой из операторов многокритериального ЭА. Для выявления эффектов, связанных со спецификой ЭА, необходимо ввести дополнительные ограничения на трудоемкость каждого из операторов многокритериального ЭА. Такой подход к сравнению ЭА и ДП реализован в следующей теореме. Обозначим через T_{DP} трудоемкость алгоритма ДП, соответствующего схеме алгоритма 4.1.

Теорема 4.1. *Пусть задача* Π' *разрешима алгоритмом ДП, условия С.1, С.2 выполнены, а обработка состояния в ДП имеет трудоемкость* $\Theta(1)$. *Тогда существует многокритериальный ЭА, где трудоемкость каждого оператора есть* $O(1)$, *и ПМА задачи* Π' *вычисляется в среднем за время* $O(T_{\mathrm{DP}}\bar{n} \log W_{\max})$.

Доказательство. Согласно утверждению 4.3 общее время счета ДП составляет

$$T_{\mathrm{DP}} = \Theta\left(\sum_{i=0}^{\bar{n}-1} |\mathcal{T}_i| \cdot |\mathcal{F}_{i+1}|\right). \tag{4.6}$$

По лемме 4.1 математическое ожидание числа итераций эволюционного алгоритма 4.2 до построения ПМА есть

$$O\left(\bar{n}\cdot\log W_{\max}\cdot\sum_{i=0}^{\bar{n}-1}|\mathcal{T}_i|\cdot|\mathcal{F}_{i+1}|\right),$$

а значит, ПМА вычислимо данным ЭА в среднем за время

$$O\left(\bar{n}\log W_{\max}\cdot\sum_{i=0}^{\bar{n}-1}(|\mathcal{T}_i|\cdot|\mathcal{F}_{i+1}|)\right)=O(\bar{n}T_{\mathrm{DP}}\log W_{\max}).$$

□

Подобный результат имеет место и в более общем случае, когда время обработки состояния ДП и трудоемкости операторов ЭА полиномиально ограничены. Как показывает следующее утверждение в этом случае существует верхняя оценка средней трудоемкости многокритериального ЭА, отличающаяся от трудоемкости алгоритма динамического программирования на множитель, ограниченный полиномом от длины входных данных и числа состояний ДП.

Утверждение 4.4. *Пусть задача* Π' *разрешима алгоритмом ДП, условия С.1, С.2 выполнены, а обработка состояния в ДП имеет полиномиально ограниченную трудоемкость. Тогда существует многокритериальный ЭА, в котором каждый оператор полиномиально вычислим, и ПМА задачи* Π' *в среднем вычисляется за время* $T_{\mathrm{DP}}\cdot\mathrm{poly}(|I|)\bar{n}\log W_{\max}$, *где* $\mathrm{poly}(\cdot)$ *– некоторый полином.*

Утверждение 4.4 доказывается аналогично теореме 4.1.

Во многих случаях применения ДП предпорядок $\preceq_{\mathrm{dom}}$ представляет собой произведение линейных порядков (см. примеры выше). В таких случаях множество вычисленных состояний ДП целесообразно хранить в таблице, где число строк равно числу линейных порядков, а число столбцов

равно $\bar{n}+1$. Для алгоритмов ДП с таким табличным представлением состояний в [35, 36] предложена многокритериальная эволюционная стратегия с оценкой среднего числа итераций, аналогичной оценке леммы 4.1.

4.2.3. Примеры применения эволюционного алгоритма на базе динамического программирования

Задача коммивояжера. По лемме 4.1 математическое ожидание числа итераций эволюционного алгоритма 4.2 на основе ДП М. Хельда и Р. Карпа есть $O\big(|V|^4 \cdot 2^{|V|}\big)$.

Задача об одномерном булевом рюкзаке. Рассмотрим эволюционный алгоритм 4.2 на основе приведенного ранее ДП трудоемкости $O(nA)$. Математическое ожидание числа итераций до получения оптимального решения в данном ЭА есть $O\big(n^2 A \log(A)\big)$ по лемме 4.1.

Задача о кратчайшем пути между двумя вершинами. Применение леммы 4.1 к ЭА, основанному на ДП для задачи о кратчайшем пути между двумя вершинами, показывает, что среднее число итераций эволюционного алгоритма 4.2 до получения оптимума есть $O\big(|V|^4 \log|V|\big)$. Эта оценка может быть снижена до $O(|V|^3 \log|V|)$ при некоторой модификации отношения доминирования в ЭА (см. [155]). Дальнейшее улучшение оценки до $O(|V|^3)$ было достигнуто в [279] при рассмотрении многокритериального варианта эволюционной стратегии.

4.2.4. Заключение

В данном разделе исследованы возможности выбора функции представления решений для многокритериального ЭА таким образом, чтобы данный алгоритм доставлял решения того же качества, что и метод динамического программирования. На основе общей схемы ДП предложен

подход к построению ЭА с возможностями алгоритмов динамического программирования и исследована средняя трудоемкость предложенных алгоритмов в сравнении с трудоемкостью ДП.

В соответствии с терминологией У. Бертеле и Ф. Бриоши в настоящем и предыдущем разделах рассматривается *сериальное ДП*. Представляет интерес возможность обобщения полученных результатов на алгоритмы *несериальноого ДП* [131]. Кроме того, было бы интересно обобщение этих результатов на случай, когда переходные функции имеют более одного аргумента (см., например, [104, 188]).

Открытым остается вопрос о возможностях распараллеливания эволюционных алгоритмов, построенных на основе ДП и их сравнении с параллельными алгоритмами динамического программирования (см. [188]).

Общая схема эволюционного алгоритма на основе динамического программирования разработана автором совместно с Б. Доэрром, Ф. Нейманном, К. Тиссеном и М. Тиле на базе подхода из [35]. При этом соавторами было предложено использовать многокритериальноый ЭА вместо многокритериальной эволюционной стратегии (см. [35]), что позволило снизить общую трудоемкость ЭА, и приведены иллюстративные примеры. К. Тиссеном доказано вспомогательное утверждение 4.2.

4.3. Вполне полиномиальная рандомизированная аппроксимационная схема

Покажем, что для многих задач комбинаторной оптимизации П описанный выше подход позволяет находить допустимые решения со сколь угодно высокой точностью при помощи ЭА.

В [293] Г. Воегингером предложены условия, при которых для достаточно широкого класса задач на основе уже известного алгоритма ДП может быть построена вполне полиномиальная аппроксимационная схема. В этот класс, например, входит задача о булевом одномерном рюкзаке и

такие задачи теории расписаний, как задача минимизации средневзвешенного времени завершения работ на фиксированном числе параллельных машин разной производительности, задача минимизации средневзвешенного отклонения от заданного срока завершения работ на одной машине, задача минимизации взвешенного суммарного запаздывания деталей и др.

Здесь для класса задач, удовлетворяющих условиям Г. Воегингера, на основе ЭА из раздела 4.2 будет построено семейство алгоритмов, являющееся *вполне полиномиальной рандомизированной аппроксимационной схемой*. Последняя определяется следующим образом (см., например, [222]).

Определение 4.1. *Под вполне полиномиальной* рандомизированной *аппроксимационной схемой подразумевается семейство вероятностных алгоритмов при всевозможных* $0 < \varepsilon < 1$ *с временной сложностью, полиномиально зависящей от длины входа задачи и от* $1/\varepsilon$*, дающих* $(1+\varepsilon)$*-приближенное решение с вероятностью не менее 3/4.*

Заметим, что вероятность отсутствия $(1+\varepsilon)$-приближенного решения с ростом числа независимых испытаний вероятностного алгоритма убывает экспоненциально, поэтому вместо $3/4$ в данном случае может использоваться любая константа из интервала (0,1).

Сделанное в начале главы предположение о сводимости задачи Π к Π' означает, что ПМА $\mathcal{S}^*$ индивидуальной задачи I' из Π' может быть эффективно преобразовано в оптимальное решение индивидуальной задачи I из Π. В настоящем разделе введем ряд дополнительных «технических» предположений относительно данной сводимости.

1. Пусть всякое $S \in \mathcal{S}^*$ эффективно отображается в решение $\mathbf{y}(S) \in \mathrm{Sol}(I)$. Полиномиально вычислимая функция $f_I(\mathbf{y}(S))$ далее обозначается через $\mathcal{G}(S)$.

2. Упрощенный алгоритм ДП решает задачу Π', поэтому имеет место

равенство

$$f_I^* = \min\{\mathcal{G}(S) : S \in \mathcal{S}_{\bar{n}}\}, \tag{4.7}$$

если П – задача минимизации, или

$$f_I^* = \max\{\mathcal{G}(S) : S \in \mathcal{S}_{\bar{n}}\}, \tag{4.8}$$

если П – задача максимизации.

3. Относительно структуры входных данных задачи П предположим, что вход I состоит из $\bar{n}$ векторов $\mathcal{X}_1, \ldots, \mathcal{X}_{\bar{n}} \in \mathbb{Z}_+^{\alpha}$ и целочисленные компоненты $x_{1k}, \ldots, x_{\alpha k}$ каждого из векторов $\mathcal{X}_k$ представлены в двоичной кодировке. Размерность α может зависеть от конкретного входа задачи. Таким образом, длина входа $|I|$ составляет $\Theta\left(\bar{n} + \alpha + \log\left(\sum_{k=1}^{\bar{n}} \sum_{i=1}^{\alpha} x_{ik}\right)\right)$.

4. При каждом i, $i = 1, \ldots, \bar{n}$, каждая функция $F \in \mathcal{F}_i$ и соответствующая ей функция H_F не зависят от векторов входа $\mathcal{X}_k$ с номерами $k \neq i$ и каждый элемент множества $\mathcal{S}'$ – вектор вида $S = (s_1, \ldots, s_{\beta}) \in \mathbb{Z}_+^{\beta}$. Размерность β фиксирована и не зависит от индивидуальной задачи I.

Предположение о том, что элементы множества $\mathcal{S}'$ суть целочисленные векторы, далее будет использовано по существу в связи с тем, что каждый компонент состояния будет представлять собой некоторый количественный параметр, который может масштабироваться. В ряде случаев условие целочисленности может быть ослаблено и рассматриваемый метод может быть применен для построения вполне полиномиальных аппроксимационных схем к ДП, где состояния имеют рациональные компоненты (см., например, [144]).

Достаточные условия существования вполне полиномиальной аппроксимационной схемы. При построении вполне полиномиальных

аппроксимационных схем на каждой стадии ДП могут исключаться некоторые состояния, близкие к уже построенным. Одной из первых работ, где использовался такой *метод «прореживания» множества состояний*, является [215]. Другие примеры его использования могут быть найдены в [60, 144, 293].

Для формализации понятия близости состояний, следуя [293], предположим, что задан *вектор степеней* $D = (d_1, \ldots, d_\beta) \in \mathbb{Z}_+^\beta$, не зависящий от индивидуальной задачи I. При заданном вещественном масштабирующем множителе $\Delta > 1$ будем полагать, что состояние $S = (s_1, \ldots, s_\beta)$ является (D, Δ)-*близким* к состоянию $S' = (s'_1, \ldots, s'_\beta)$, если

$$\Delta^{-d_\ell} s_\ell \leq s'_\ell \leq \Delta^{d_\ell} s_\ell, \quad \ell = 1, \ldots, \beta.$$

Введем обозначение $\mathcal{L}_0$ для множества индексов $1 \leq \ell \leq \beta$, таких что $d_\ell = 0$, и пусть $\mathcal{L}_1 = \{1, \ldots, \beta\} \backslash \mathcal{L}_0$.

Основным средством исключения «неперспективных» состояний во вполне полиномиальной аппроксимационной схеме на основе ДП [293] является отношение линейного предпорядка $\preceq_{\text{qua}}$, заданное на множестве состояний. При «прореживании» на каждой стадии i, $i = 1, \ldots, \bar{n}$, могут удаляться только те из состояний, которые уступают какому-либо уже построенному на этой стадии состоянию в смысле отношения $\preceq_{\text{qua}}$ и при этом (D, Δ)-близки к нему.

Линейный предпорядок $\preceq_{\text{qua}}$ выбирается таким образом, чтобы он был расширением предпорядка $\preceq_{\text{dom}}$, т. е. если $S \preceq_{\text{dom}} S'$, то $S \preceq_{\text{qua}} S'$ при любых $S, S' \in \mathbb{Z}_+^\beta$. Для совместимости с результатами [293] введем более ограничительное, чем в разделах 4.1 и 4.2, предположение о том, что отношение $\preceq_{\text{dom}}$ является частичным порядком. Данное ограничение не является существенным с точки зрения приложений ДП.

Заметим, что для индивидуальной задачи I любой частичный порядок $\preceq_{\text{dom}}$ на семействе конечных подмножеств $\mathcal{S}_1, \ldots, \mathcal{S}_{\bar{n}}$ может быть представлен с помощью конечного числа критериев $g_1, \ldots, g_{d'}$ в соответствую-

щей I индивидуальной задаче P' так, чтобы отношение доминирования по Парето было эквивалентным отношению $\preceq_{\mathrm{dom}}$ на этом множестве (см., например, [3], гл. 2, п. 4). Таким образом, для совместимости с подходом из разделов 4.1 и 4.2 может быть указана соответствующая $\preceq_{\mathrm{dom}}$ задача Π'.

Сформулируем условия [293], обеспечивающие существование вполне полиномиальной аппроксимационной схемы. В дополнение к С.1 и С.2 введем следующие условия С.1′, С.2′, С.3 и С.4.

Условие С.1′. При любых $\Delta > 1$, $S, S' \in \mathbb{Z}_+^\beta$ и $F \in \mathcal{F}_i$, $i = 1, \ldots, \bar{n}$, если S является (D, Δ)-близким к S', $F(S') \in \mathcal{S}$ и $S \preceq_{\mathrm{qua}} S'$, то либо $F(S) \preceq_{\mathrm{qua}} F(S')$ и $F(S)$ является (D, Δ)-близким к $F(S')$, либо $F(S) \preceq_{\mathrm{dom}} F(S')$.

Условие С.2′. При $\Delta > 1$, $S, S' \in \mathbb{Z}_+^\beta$, $i = 1, \ldots, \bar{n}$ и $F \in \mathcal{F}_i$, если состояния S и S' являются (D, Δ)-близкими и $S \preceq_{\mathrm{qua}} S'$, то $H_F(S') \leq H_F(S)$.

Условие С.3. Существует величина $\gamma \in \mathbb{Z}_+$, зависящая только от функции $\mathcal{G}$ и вектора D, такая что для любых $\Delta > 1$ и $S, S' \in \mathbb{Z}_+^\beta$,

(i) если S является (D, Δ)-близким к S' и $S \preceq_{\mathrm{qua}} S'$, то $\mathcal{G}(S') \leq \Delta^\gamma \mathcal{G}(S)$ в случае задачи минимизации и $\Delta^{-\gamma} \mathcal{G}(S) \leq \mathcal{G}(S')$ в случае задачи максимизации;

(ii) если $S \preceq_{\mathrm{dom}} S'$, то $\mathcal{G}(S') \leq \mathcal{G}(S)$ в случае задачи минимизации и $\mathcal{G}(S') \geq \mathcal{G}(S)$ в случае задачи максимизации.

Условие С.4.

(i) Все функции $F \in \mathcal{F}_i$ и H_F при любом $i = 1, \ldots, \bar{n}$, функция $\mathcal{G}$, а также отношение $\preceq_{\mathrm{qua}}$ вычислимы за полиномиальное время.

(ii) Мощности $|\mathcal{F}_i|,\ i = 1, \ldots, \bar{n}$, полиномиально ограничены.

(iii) Множество $\mathcal{S}_0$ вычислимо за полиномиально ограниченное время.

(iv) Существует полином $\pi_1(\bar{n}, \log_2 |I|)$, такой что компоненты состояний множеств $\mathcal{S}_i,\ i = 1, \ldots, \bar{n}$, суть целые неотрицательные числа, не превышающие $e^{\pi_1(\bar{n}, \log_2 |I|)}$. Кроме того, для всех $\ell \in \mathcal{L}_0$ мощность множества значений, которые может принимать координата s_ℓ вектора $S \in \mathcal{S}_i$, ограничена некоторым полиномом $\pi_2(\bar{n}, \log_2 |I|)$, т. е. $|\{s_\ell : (s_1, \ldots, s_\ell, \ldots, s_\beta) \in \mathcal{S}_i\}| \leq \pi_2(\bar{n}, \log_2 |I|)$.

Пусть при условиях, сформулированных выше в настоящем разделе, существуют частичный порядок $\preceq_{\mathrm{dom}}$, линейный предпорядок $\preceq_{\mathrm{qua}}$ и вектор степеней D, удовлетворяющие условиям С.1, С.2, С.1′, С.2′, С.3 и С.4. Тогда в соответствии с терминологией из [293] задача П является *DPB-задачей* (от английского термина DP-benevolent problem).

Вполне полиномиальная аппроксимационная схема. Рассмотрим алгоритм $\mathrm{ДП}_\Delta$, представляющий собой модификацию ДП с «прореживанием» множества состояний на каждой стадии. Для определения наборов состояний, являющихся (D, Δ)-близкими между собой, в алгоритме $\mathrm{ДП}_\Delta$ используется разбиение множества состояний на выбранные специальным образом параллелепипеды (определение будет дано ниже). Такое разбиение позволяет исключать из рассмотрения некоторые «близкие» к уже имеющимся состояния.

Пусть L – достаточно большая величина, выбранная для данной индивидуальной задачи I и погрешности ε (выбор величины L будет обсуждаться далее). При описании алгоритма $\mathrm{ДП}_\Delta$ будем пользоваться разбиением множества $P(L, \Delta) = \mathbb{Z}_+^\beta \cap [0, \Delta^L]^\beta$ на семейство параллелепипедов

$$\{\mathcal{P}_{(k_1, \ldots, k_\beta)} : k_\ell = 0, \ldots, L,\ \ell = 1, \ldots, \beta\},$$

где параллелепипед $\mathcal{P}_{(k_1,\ldots,k_\beta)}$ с индексом $(k_1,\ldots,k_\beta) \in \mathbb{Z}_+^\beta$ содержит все целочисленные точки $S = (s_1,\ldots,s_\beta)$, удовлетворяющие условиям:

$$s_\ell \in \begin{cases} \{0\}, & \text{если } k_\ell = 0; \\ [\Delta^{k_\ell - 1}, \Delta^{k_\ell} - 1], & \text{если } 0 < k_\ell < L; \\ [\Delta^{k_\ell - 1}, \Delta^{k_\ell}], & \text{если } k_\ell = L \end{cases} \tag{4.9}$$

при $\ell \in \mathcal{L}_1$, и

$$s_\ell = k_\ell \tag{4.10}$$

при $\ell \in \mathcal{L}_0$.

Алгоритм 4.3. ДП$_\Delta$

1. Инициализировать $\mathcal{T}_0 := \mathcal{S}_0$.
2. Для i от 1 до $\bar{n}$ выполнять:
3. Положить $\mathcal{T}_i := \emptyset$.
4. Для каждого $S \in \mathcal{T}_{i-1}$ и каждого $F \in \mathcal{F}_i$ выполнять:
5. Пусть $F(S)$ содержится в некотором $\mathcal{P}_{(k_1,\ldots,k_\beta)}$.
6. Если $H_F(S) \le 0$ и в параллелепипеде $\mathcal{P}_{(k_1,\ldots,k_\beta)}$ не существует такого $S' \in \mathcal{T}_i$, что $F(S) \preceq_{\text{qua}} S'$, то добавить $F(S)$ в $\mathcal{T}_i$ и удалить из $\mathcal{T}_i$ все состояния $S'' \in \mathcal{P}_{(k_1,\ldots,k_\beta)} \cap \mathcal{T}_i$, такие что $S'' \preceq_{\text{qua}} F(S)$.
7. Конец цикла по S и F.
8. Конец цикла по i.
9. Выбрать $\min\{\mathcal{G}(S) : S \in \mathcal{T}_{\bar{n}}\}$ в случае задачи минимизации либо $\max\{\mathcal{G}(S) : S \in \mathcal{T}_{\bar{n}}\}$ в случае задачи максимизации.

Как показано в [293], для получения вполне полиномиальной аппроксимационной схемы в ДП$_\Delta$ достаточно положить

$$\Delta = 1 + \frac{\varepsilon}{2\gamma\bar{n}}, \tag{4.11}$$

$$L = \left\lceil \frac{\pi_1(\bar{n}, \log_2 |I|)}{\ln \Delta} \right\rceil . \tag{4.12}$$

При этом, как вытекает из следующей леммы, L полиномиально ограничена от длины входа и от $1/\varepsilon$.

Лемма 4.2. [293] *Пусть* $\Delta = 1+\varepsilon/(2\gamma\bar{n})$, *тогда величина* $L = \left\lceil \frac{\pi_1(\bar{n},\log_2 |I|)}{\ln \Delta} \right\rceil$ *полиномиально ограничена относительно длины входа и от* $1/\varepsilon$ *и всякое состояние из множества* $\mathcal{S}_i$, $i = 1, \ldots, \bar{n}$, *принадлежит одному из параллелепипедов* $\mathcal{P}_{(i_1,\ldots,i_\beta)}$, *заданных условиями (4.9) и (4.10).*

Основной результат [293] формулируется следующим образом.

Теорема 4.2. *Пусть* Π – *DPB-задача. Тогда семейство алгоритмов* $ДП_\Delta$ *с выбором параметров* Δ *и* L *согласно (4.11) и (4.12) представляет собой вполне полиномиальную аппроксимационную схему.*

4.3.1. Эволюционный алгоритм

Предлагаемый здесь эволюционный алгоритм EA_Δ аналогичен эволюционному алгоритму 4.2. Основной цикл алгоритма отличается лишь правилом добавления особей в популяцию, что выражается в замене отношения $\preceq_{\mathrm{EA}}$ на новое отношение $\preceq_\Delta$, где $(i, S) \preceq_\Delta (i', S')$ тогда и только тогда, когда $i = i'$ и существует такой набор индексов $k_1, \ldots, k_\beta$, что $S, S' \in \mathcal{P}_{(k_1,\ldots,k_\beta)}$, причем $S \preceq_{\mathrm{qua}} S'$.

Алгоритм EA_Δ действует в предположении, что Π является DPB-задачей. Работа начинается с исходной популяции X, такой что $\hat{X} = \{0\}\times\mathcal{S}_0$, для построения которой используется полиномиальная процедура, существующая согласно условию C.4 (iii).

Из схемы алгоритма 4.2 вытекает, что для любого $i \in \{1, \ldots, \bar{n}\}$ особи вида (i, S) в популяции EA_Δ представляют только состояния $S \in \mathcal{S}_i$, так как состояния из $\mathbb{Z}^\beta \backslash \mathcal{S}_i$ не могут быть построены с помощью операторов $\mathrm{Sel}^\diamond$ и $\mathrm{Mut}^\diamond$ за i итераций.

4.3.2. Вполне полиномиальная рандомизированная аппроксимационная схема на основе эволюционного алгоритма

Без потери общности в настоящем разделе будем полагать, что Π – задача минимизации. Для произвольного $S \in \mathcal{S}_i$ обозначим через $\theta(i, S)$ случайную величину, равную номеру итерации EA_Δ, на которой впервые получена особь (i, S'), такая что S' (D, Δ^i)-близко к S и $S \preceq_{\mathrm{qua}} S'$. Заметим, что по схеме EA_Δ, начиная с итерации $\theta(i, S)$, в популяции всегда можно найти некоторую особь, удовлетворяющую данному условию.

Приведем результат из [293], который будет использован в лемме 4.4.

Лемма 4.3. [293] *Пусть S – недоминируемое состояние в $\mathcal{S}_i$ для некоторого $1 \leq i \leq \bar{n}$, тогда для S найдется недоминируемое в $\mathcal{S}_{i-1}$ состояние $S^\#$ и отображение $F^\# \in \mathcal{F}_i$, такие что $F^\#(S^\#) = S$.*

Следующая лемма показывает, что для любого недоминируемого $S \in \mathcal{S}_i$ в среднем за полиномиально ограниченное от $|I|$ и $1/\varepsilon$ число итераций обнаруживается особь (i, S'), такая что S' (D, Δ^i)-близко к S и $S \preceq_{\mathrm{qua}} S'$.

Лемма 4.4. *Пусть Π является DPB-задачей. Тогда для любого $i = 0, \ldots, \bar{n}$ и всякого недоминируемого в $\mathcal{S}_i$ состояния S выполняется неравенство $\mathbf{E}[\theta(i, S)] \leq \bar{n}(L\pi_2(\bar{n}, \log_2 |I|))^\beta \cdot \sum_{k=1}^{i} |\mathcal{F}_k|$.*

Доказательство. Воспользуемся индукцией по i. При $i = 0$ утверждение выполняется тривиально. Рассмотрим произвольное недоминируемое состояние S в $\mathcal{S}_i$. Пусть $i > 0$ и утверждение леммы верно для $i - 1$.

Относительно алгоритма ДП доказано (см. лемму 4.3), что для S найдутся недоминируемое в $\mathcal{S}_{i-1}$ состояние $S^\#$ и отображение $F^\# \in \mathcal{F}_i$, такие что $F^\#(S^\#) = S$. Отметим, что предположение индукции дает оценку сверху для математического ожидания числа итераций $\theta(i-1, S^\#)$ до получения особи $(i-1, U^\#)$, такой что $U^\#$ (D, Δ^{i-1})-близко к $S^\#$ и $S^\# \preceq_{\mathrm{qua}} U^\#$. Назовем мутацию с применением $F^\#$ к особи $(i-1, U^\#)$ *успешной*.

Согласно условию C.2′ имеем:

$$H_{F^\#}(U^\#) \le H_{F^\#}(S^\#) \le 0,$$

а по предположению C.1′ либо (а) $F^\#(U^\#)$ (D, Δ^{i-1})-близко к S и $S \preceq_{\text{qua}} F^\#(U^\#)$, либо (б) $S \preceq_{\text{dom}} F^\#(U^\#)$.

В случае (а) после успешной мутации популяция будет содержать элемент $(i, F^\#(U^\#))$ либо некоторый другой элемент (i, U'), такой что U' попадает в тот же параллелепипед $\mathcal{P}_{(k_1,\ldots,k_\beta)}$, что и $F^\#(U^\#)$, и кроме того, $U' \succeq_{\text{qua}} F^\#(U^\#)$. То есть после указанной мутации популяция будет содержать особь (i, U), такую что U (D, Δ)-близко к $F^\#(U^\#)$ и $F^\#(U^\#) \preceq_{\text{qua}} U$. Далее так как $F^\#(U^\#)$ (D, Δ^{i-1})-близко к S, то из определения близости состояний вытекает, что U (D, Δ^i)-близко к S. Кроме того, $S \preceq_{\text{qua}} F^\#(U^\#) \preceq_{\text{qua}} U$, следовательно, $S \preceq_{\text{qua}} U$. То есть в случае (а) успешная мутация обеспечивает присутствие искомого представителя для S в популяции на шаге индукции i.

В случае (б) после успешной мутации будет получен элемент (i, S), так как S – недоминируемое состояние, а $F^\#(U^\#) \succeq_{\text{dom}} S$. В результате популяция будет содержать элемент (i, U), такой что U (D, Δ)-близко к S, и кроме того, $U \succeq_{\text{qua}} S$. Очевидно, при этом U также (D, Δ^i)-близко к S. Таким образом, на шаге индукции i успешная мутация обеспечивает присутствие искомого представителя для S в популяции.

Для завершения доказательства остается оценить математическое ожидание числа испытаний θ^* до осуществления успешной мутации при условии наличия особи $\xi^\# = (i-1, U^\#)$ в текущей популяции. Заметим, что по схеме алгоритма вероятность успешной мутации $p^* = \left(\bar{n} \cdot |\{(i-1, S') \in \hat{X}\}| \cdot |\mathcal{F}_i|\right)^{-1}$, при этом

$$|\hat{X}| = \sum_{i'=1}^{\bar{n}} |\{(i', S') \in \hat{X}\}| \le \sum_{i'=1}^{\bar{n}} |\{(k_1, \ldots, k_\beta) : \mathcal{P}_{(k_1,\ldots,k_\beta)} \cap \mathcal{S}_{i'} \ne \emptyset\}|. \quad (4.13)$$

Рассмотрим слагаемое в правой части (4.13) при любом фиксированном i'. Для каждого $\ell \in \mathcal{L}_1$ индекс k_ℓ принимает не более чем $L{+}1$ значений.

Кроме того, ввиду условия C.4 (iv) для каждого $\ell' \in \mathcal{L}_0$ координата $s_{\ell'}$ в состояниях из $\mathcal{S}_{i'}$ может принимать не более $\pi_2(\bar{n}, \log_2 |I|)$ значений.

Таким образом, правая часть неравенства (4.13) не превышает $\bar{n}L^{|\mathcal{L}_1|}\pi_2(\bar{n}, \log_2 |I|)^{|\mathcal{L}_0|} \leq \bar{n}(L\pi_2(\bar{n}, \log_2 |I|))^{\beta}$, а значит, $\mathbf{E}[\theta^*] = 1/p^* \leq \bar{n}(L\pi_2(\bar{n}, \log_2 |I|))^{\beta}|\mathcal{F}_i|$. Утверждение леммы для шага индукции i вытекает из того, что $\mathbf{E}[\theta(i, S)] = \mathbf{E}[\theta(i-1, U^{\#})] + 1/p^*$. □

Полученная в лемме 4.4 оценка величины $\mathbf{E}[\theta(i, S)]$ служит основой для выбора критерия остановки алгоритма EA_{Δ}. Положим

$$\tau = 4\bar{n}(L\pi_2(\bar{n}, \log_2 |I|))^{\beta} \cdot \sum_{i=1}^{\bar{n}} |\mathcal{F}_i|. \tag{4.14}$$

Теорема 4.3. *Если* П *является DPB-задачей, то семейство алгоритмов* EA_{Δ} *с выбором параметров* Δ *и* L *согласно (4.11) и (4.12) и критерием остановки (4.14) образует вполне полиномиальную рандомизированную аппроксимационную схему.*

Доказательство. Рассуждения аналогичны доказательству основного результата в [293]. Согласно сделанным предположениям $f_I^* = \min\{\mathcal{G}(S) : S \in \mathcal{S}_{\bar{n}}\}$. Поэтому, ввиду условия C.3 (ii), существует недоминируемое состояние $S^* \in \mathcal{S}_{\bar{n}}$, такое что $f_I^* = \mathcal{G}(S^*)$. По лемме 4.4 в среднем не более чем за $\bar{n}(L\pi_2(\bar{n}, \log_2 |I|))^{\beta} \sum_{i=1}^{\bar{n}} |\mathcal{F}_i|$ итераций в EA_{Δ} вычисляется популяция, в которой найдется особь $(\bar{n}, T^*)$, такая что T^* $(D, \Delta^{\bar{n}})$-близко к S^* и $S^* \preceq_{qua} T^*$. Тогда по условию C.3 (i):

$$\mathcal{G}(T^*) \leq \Delta^{\gamma\bar{n}}\mathcal{G}(S^*) = \left(1 + \frac{\varepsilon}{2\gamma\bar{n}}\right)^{\gamma\bar{n}} f_I^* \leq (1+\varepsilon)f_I^*.$$

(Последнее неравенство вытекает из того, что $\gamma\bar{n} \geq 1$, поэтому $(1 + \frac{\varepsilon}{2\gamma\bar{n}})^{\gamma\bar{n}}$, как функция от ε, на отрезке $\varepsilon \in [0, 2]$ выпукла и неравенство выполнено при $\varepsilon \in \{0, 2\}$.) В таком случае алгоритм обеспечивает требуемую точность по целевой функции и с помощью процедуры обратного хода по T^* может быть эффективно получено $(1+\varepsilon)$-приближенное решение.

При выполнении EA_Δ с критерием остановки (4.14), согласно неравенству Маркова, вероятность *не найти* $(1+\varepsilon)$-приближенное решение составляет не более $1/4$, как и требуется в определении вполне полиномиальной рандомизированной аппроксимационной схемы.

По условию C.4 трудоемкость итерации EA_Δ полиномиально ограничена от $|I|$ и $1/\varepsilon$, поэтому данное семейство алгоритмов является вполне полиномиальной рандомизированной аппроксимационной схемой.

Для задач максимизации доказательство аналогично. □

4.3.3. Заключение

Из полученных результатов следует, что к эволюционным алгоритмам наряду со сходимостью к оптимуму почти наверное (см., например, [277, 278]) естественно предъявить новые требования в терминах среднего времени получения искомого приближенного решения за полиномиальное время.

В условиях настоящего раздела верхние оценки среднего времени первого достижения оптимального значения целевой функции в ЭА, основанном на ДП, могут быть снижены за счет исключения логарифмического множителя. Для получения улучшенных оценок достаточно вместо леммы 4.1 воспользоваться рассуждениями аналогичными доказательству леммы 4.4.

5. Применение оптимальной рекомбинации в генетических алгоритмах

5.1. Задача о наименьшем покрытии

Рассматривается задача наименьшего покрытия (ЗНП), формулировка которой приведена в разделе 1.1. Во многих случаях предпочтительнее рассматривать ЗНП как задачу ЦЛП: найти

$$\max \mathbf{cx} \tag{5.1}$$

при условиях

$$\mathbf{Qx} \geq \mathbf{e}, \tag{5.2}$$

$$\mathbf{x} \in \{0,1\}^n, \tag{5.3}$$

где $\mathbf{e}$ обозначает единичный вектор-столбец длины m; $\mathbf{Q}$ – $(m \times n)$-матрица, причем $q_{ij} = 1$, если элемент i покрывается подмножеством M_j (т. е. $i \in M_j$), и $q_{ij} = 0$ в противном случае. Пользуясь этой формулировкой, ЗНП можно рассматривать как задачу оптимального покрытия строк матрицы $\mathbf{Q}$ ее столбцами, где столбцы соответствуют множествам M_j. Будем предполагать, что эти множества упорядочены по невозрастанию c_j.

Для нахождения точного решения ЗНП в литературе предложены алгоритмы, основанные на вычислительных схемах методов ветвей и границ, отсечений, перебора L-классов и др. [40, 52, 63, 120]. Плодотворным оказался гибридный подход, в частности, комбинации метода ветвей и границ с лагранжевой релаксацией [119] и отсечениями [120, 128] позволили значительно повысить размерность задач, решаемых за приемлемое время.

Наряду с точными методами для ЗНП разработан ряд приближенных алгоритмов полиномиальной трудоемкости с гарантированными оценками

точности (см., например, [83, 146]). В то же время отыскание полиномиального алгоритма с гарантированной оценкой точности $(1-\varepsilon)\ln m$ при $\varepsilon > 0$ представляется проблематичным (см. ссылки в [40]).

Одним из первых приближенных алгоритмов, предложенных для невзвешенной ЗНП, является жадный алгоритм. В этом алгоритме на каждой итерации в качестве очередного элемента покрытия выбирается подмножество, покрывающее наибольшее число элементов из M, не покрытых на предыдущих итерациях. В [83] показано, что если каждая строка в $\mathbf{Q}$ содержит не менее k единиц, то мощность получаемого покрытия оценивается сверху величиной $1 + \frac{n}{k}\left(1 + \ln\frac{mk}{n}\right)$. Данное неравенство дает также верхнюю оценку мощности оптимального покрытия для указанного класса матриц. Неалгоритмические верхние оценки мощности наименьших покрытий для некоторых классов невзвешенных задач получены в [77, 92, 99].

Для поиска приближенных решений задач наименьшего покрытия с большой размерностью широкое распространение получили эвристические алгоритмы, в частности, методы лагранжевой релаксации [141], генетические алгоритмы [112, 127], алгоритмы муравьиной колонии [111], нейронные сети [205] и другие. Большинство из них основано на использовании эффективных эвристических процедур с учетом некоторой статистической информации, накопленной в процессе решения задачи.

5.1.1. Генетический алгоритм с недвоичным представлением

Достаточно эффективный вариант ГА для ЗНП предложен Дж. Бисли и П. Чу (J. Beasley, P. Chu) в работе [127]. Этот алгоритм использует двоичное представление решений в виде строк из нулей и единиц длины n. Столбец j при этом входит в покрытие, если j-й бит генотипа равен 1. Проблема при использовании данного представления связана с возможностью получения недопустимых решений после мутации и кроссинговера. Для восстановления допустимости в [127] применяется оператор корректи-

ровки решений, представляющий собой эвристику жадного типа.

В предлагаемом в данном разделе ГА используется недвоичное представление решений, благодаря которому обеспечивается допустимость фенотипа при декодировании любого генотипа. Обозначим через U_i множество номеров столбцов, покрывающих строку i. В ГА с недвоичным представлением (GANP) генотип ξ состоит из $l = m$ генов $\xi_1, \xi_2, ..., \xi_m$, причем $\xi_i \in U_i$, где $i = 1, 2, ..., m$, т. е. каждый ген кодирует один из столбцов, покрывающих соответствующую строку. Генотипу ξ сопоставляется вектор-индикатор покрытия $\mathbf{x}(\xi)$, где $x_i(\xi) = 1$ при условии, что столбец j представлен хотя бы в одном из генов генотипа ξ. В таком случае нет необходимости в операторе корректировки, однако для учета специфики задачи в данном ГА требуется оператор локальной корректировки решений, исключающий после кроссинговера и мутации некоторые избыточные столбцы. С этой целью в предложенном алгоритме используются жадные эвристики, вычисляющие приближенное решение вспомогательной задачи наименьшего покрытия строк матрицы $\mathbf{Q}$ посредством столбцов из множества $J(\xi) = \{j | x_j(\xi) = 1\}$, где ξ – полученный после мутации генотип.

Модифицированная особь добавляется в популяцию при условии, что там отсутствуют особи с тем же фенотипом. Генотипы начальной популяции порождаются независимо, и каждый ген ξ_i выбирается равновероятно из множества «перспективных» столбцов $U_i(\alpha) \subseteq U_i$. Здесь $U_i(\alpha)$ вводится по аналогии с подходом из [127]. Пусть $W_i,\ i = 1, ..., m$, – множество из α столбцов с наименьшей стоимостью среди покрывающих строку i. Полагаем $U_i(\alpha) = \bigcup_{k=1}^{m} (W_k \cap U_i)$, $i = 1, ..., m$. Значение α фиксировано в процессе работы ГА и выбирается с учетом свойств решаемой задачи.

Функция приспособленности $\Phi(\xi)$ на итерации t имеет вид:

$$\Phi(\xi) = \mathbf{cx}(\xi^{j(t)}) - \mathbf{cx}(\xi) + c_n,$$

где $j(t)$ – номер особи, имеющей наибольшую стоимость покрытия в популяции на данной итерации. Отбор родительских генотипов реализован с

помощью оператора пропорциональной селекции (см. п. 1.2.1).

Алгоритм GANP базируется на стационарной схеме управления популяцией. Операторы кроссинговера, мутации и три процедуры локальной корректировки решений Lmin, Grd и DGrd, используемые в предлагаемом алгоритме, будут описаны после общей схемы GANP.

Алгоритм 5.1. Алгоритм GANP

1. Положить $t := 0$.

2. Для k от 1 до N выполнять:

2.1. Построить случайным образом генотип ξ, выбирая каждый ген ξ_i равновероятно из множества $U_i(\alpha)$.

2.2. Положить $\xi^{k,0} := \mathrm{Lmin}(\xi)$.

Итерация t.

3. Выбрать особи $\xi^u := \xi^{\mathrm{Sel}(X^t),t}$, $\xi^w := \xi^{\mathrm{Sel}(X^t),t}$ с помощью пропорциональной селекции.

4. Вычислить $\xi := \mathrm{Mut}(\mathrm{Cross}(\xi^u, \xi^w))$.

5. Вычислить $\eta := \mathrm{Grd}(\xi)$, $\eta' := \mathrm{DGrd}(\xi)$ и $\Phi(\eta)$, $\Phi(\eta')$.

6. Если $\Phi(\eta) > \Phi(\eta')$, то положить $\xi' := \eta$; иначе $\xi' := \eta'$.

7. Если в X^t нет особей с фенотипом $\mathbf{x}(\xi')$, то заменить генотип с наименьшей приспособленностью в X^t на ξ'; иначе заменить генотип с наименьшей приспособленностью генотипом ξ, полученным на шаге 4.

8. Положить $X^{t+1} := X^t$, $t := t + 1$.

9. Если условие остановки не выполнено, то идти на шаг 3.

10. Результатом работы является особь с наибольшей приспособленностью среди найденных за все итерации.

Операторы кроссинговера

В ГА для ЗНП предложен новый оператор кроссинговера, называемый *ЛП-кроссинговером* [173]. При действии данного оператора решение-потомок вычисляется с использованием методов линейного программирования. Далее рассматриваются два варианта данного оператора: ЛП-кроссинговер с решением ЗОР (см. определение 3.1) и ЛП-кроссинговер с решением ослабленной ЗОР (см. условие (3.2)).

По умолчанию через GANP обозначается вариант ГА с использованием ЛП-кроссинговера, основанного на решении ослабленной ЗОР. В случае использования ЛП-кроссинговера, основанного на решении ЗОР, или равномерного кроссинговера (см. п. 1.2.1), это специально уточняется.

Для описания оператора ЛП-кроссинговера сформулируем вспомогательную задачу, которая состоит в отыскании наименьшего покрытия множества M посредством покрывающих подмножеств с индексами из $U' = J(\xi^u) \cup J(\xi^w)$, что соответствует условию (3.2) ослабленной ЗОР. Требуется найти

$$\min_{K \subseteq U'} \sum_{j \in K} c_j \tag{5.4}$$

при условии

$$\bigcup_{j \in K} M_j = M. \tag{5.5}$$

К задаче (5.4), (5.5) применяется следующая процедура упрощения. Обозначим $S = \{i \in M \mid |U_i \cap U'| = 1\}$. Тогда каждая строка с индексом $i \in S$ в рассматриваемой подзадаче может быть покрыта единственным столбцом $j(i)$. Пусть $R = \{j \mid j = j(i), i \in S\}$ – множество фиксированных столбцов. Каждому гену ξ_i, где $i \in \bigcup_{j \in R} M_j$, назначается индекс некоторого столбца из множества R, покрывающего строку i. Таким образом, задача (5.4), (5.5) сводится к задаче наименьшего покрытия нефиксированными столбцами всех строк из $M \setminus \bigcup_{j \in R} M_j$.

Рассмотрим линейную релаксацию задачи наименьшего покрытия строк из $M \setminus \bigcup\limits_{j \in R} M_j$ нефиксированными столбцами, где условия целочисленности заменены неравенствами $x_j \geq 0$ для всех переменных подзадачи. Данная задача решается с помощью двойственного симплекс-метода. Если полученное решение $\mathbf{x}'$ оказывается целочисленным, то на его основе назначаются оставшиеся неопределенными в процессе упрощения задачи гены генотипа ξ, соответствующие строкам из $M \setminus \bigcup\limits_{j \in R} M_j$. Если решение $\mathbf{x}'$ содержит дробные компоненты, кроссинговер не выполняется и результатом действия оператора является копия генотипа одного из родителей.

Для предотвращения чрезмерных вычислительных затрат поиск оптимума линейной задачи отменяется, если мощность $M \setminus \bigcup\limits_{j \in R} M_j$ превышает некоторый порог μ или выполнено более установленного числа ν итераций симплекс-метода (далее в экспериментах $\mu = 150$, $\nu = 300$). В этих случаях оператор также возвращает копию генотипа одного из родителей.

Аналогично описанному оператору ЛП-кроссинговера может быть построен оператор кроссинговера с решением ЗОР, соответствующей определению 3.1. В таком случае ГА будем обозначать через GAOR.

Операторы мутации

Оператор мутации организован по схеме, аналогичной пропорциональной селекции. Он применяется к каждому гену с заданной вероятностью мутации $P_{\mathtt{m}}$. Предположим, что мутация происходит в гене ξ_i. Тогда новое значение гена выбирается из множества индексов $U_i(\alpha)$ и вероятность назначения столбца $j \in U_i(\alpha)$ равна

$$p_i(j) = \frac{\frac{1}{c_j}}{\sum\limits_{k \in U_i(\alpha)} \frac{1}{c_k}}.$$

При $\alpha \geq \max\limits_i |U_i|$ и $P_{\mathtt{m}} > 0$ в результате мутации любое наперед заданное покрытие может быть получено с ненулевой вероятностью. Пусть q_t^* –

вероятность того, что оптимум не будет найден к итерации t. Тогда при $\alpha \geq \max\limits_{i} |U_i|$ и $P_{\mathtt{m}} > 0$ для любой индивидуальной задачи существует такое $\delta < 1$, что $q_t^* \leq \delta^t$.

Рассматривались и другие варианты мутации. В частности, опробован оператор, в котором мутируемый ген ξ_i выбирается равновероятно из множестве индексов $U_i(\alpha)$, а также вариант мутации с запретом на использование тех столбцов, которые уже имеются в решении. Предлагаемый оператор в ходе экспериментов показал себя лучше других.

Локальная корректировка решений

Для исключения избыточных столбцов из покрытия, соответствующего генотипу, полученному после мутации или при построении популяции X^0, предлагается использовать следующие эвристики жадного типа.

Алгоритм поиска лексикографического минимума Lmin начинает работу с данного на вход покрытия $J(\xi)$ и, просматривая его столбцы в порядке возрастания номеров, исключает те столбцы, удаление которых не нарушает допустимости решения. Данный алгоритм представляет собой обратный жадный алгоритм и, как следует из результатов В.П. Ильева [216], является $(m-1)$-приближенным алгоритмом. Вектор покрытия, полученный в результате работы Lmin, лексикографически минимален среди допустимых решений задачи покрытия строк матрицы $\mathbf{Q}$ столбцами $J(\xi)$.

Жадный алгоритм Grd – это алгоритм подъема (см. [83, 146]), который начиная с пустого множества строит покрытие $J \subseteq J(\xi)$. Пусть M' – множество еще не покрытых строк. На каждой итерации алгоритма к покрытию J добавляется столбец, для которого отношение его стоимости к числу покрываемых строк из M' минимально. Полученное таким образом покрытие J может не быть минимальным по включению. В связи с этим к

покрытию J применяется алгоритм Lmin, обеспечивающий минимальность по включению. Результатом работы Grd является модифицированный генотип η. Как следует из оценки, полученной в [146], покрытие J не более чем в $1 + \ln m$ раз уступает по целевой функции наименьшему по стоимости покрытию, содержащемуся в $J(\xi)$. Легко видеть, что трудоемкость алгоритма Grd не превышает $O(s_\xi \cdot |J|)$, где $s_\xi = \sum_{j \in J(\xi)} |M_j|$.

Двойственный жадный алгоритм DGrd представляет собой вариант жадного алгоритма спуска с адаптивным пересчетом оценок столбцов. Пусть M' обозначает множество нефиксированных строк, J' – множество оставшихся столбцов. В начале работы алгоритма множество фиксированных строк пусто. На каждой итерации из дальнейшего рассмотрения исключается один столбец. Пусть строка с индексом i называется «критической», если она может быть покрыта только одним столбцом $k(i)$ из J'. В случае, если среди нефиксированных строк имеется «критическая» строка с номером i, всем генам, соответствующим нефиксированным строкам, покрываемым столбцом $k(i)$, присваивается индекс $k(i)$. После этого данный столбец исключается, а нефиксированные строки, покрытые им, становятся фиксированными. Если же «критических» строк не оказалось, исключается тот столбец из J', для которого максимально отношение $\frac{c_j}{|M_j \cap M'|}$.

Алгоритм 5.2. DGrd

1. Положить $U'_i := U_i \cap J(\xi)$ для $i = 1, ..., m$; $M' := M$, $J' := J(\xi)$.
2. Пока $M' \neq \emptyset$ выполнять:
   - 2.1. Если есть такие $i \in M'$, что $|U'_i| = 1$, то
     Для $k \in U'_i$ положить $h^{(j)} := k$ при всех $j \in M_k \cap M'$.
     Положить $M' := M' \backslash M_k$.
   - 2.2. Иначе выбрать $k \in J'$ такой, что $\frac{c_k}{|M_k \cap M'|} \geq \frac{c_j}{|M_j \cap M'|}$ для всех $j \in J'$, и для всех $j \in M_k \cap M'$ положить $U'_j := U'_j \backslash \{k\}$.
   - 2.3. Положить $J' := J' \backslash \{k\}$.

В результате работы DGrd получен новый генотип η, соответствующий некоторому минимальному по включению покрытию. Трудоемкость данного алгоритма не превышает $O(s_\xi \cdot |J(\xi)|)$.

При экспериментальном сравнении процедур корректировки выяснилось, что отказ от использования алгоритмов Grd или DGrd приводит к существенному понижению эффективности GANP (в смысле качества полученных решений) для многих тестовых задач. С другой стороны, при решении отдельных задач использование Grd или DGrd для корректировки особей начальной популяции в дальнейшем затрудняет поиск оптимальных решений. В связи с этим при построении начальной популяции используется алгоритм Lmin, позволяющий получить большее разнообразие особей.

5.1.2. Гибридный точный алгоритм

Разработанный выше генетический алгоритм GANP может использоваться в комбинации с точными алгоритмами типа ветвей и границ для получения начального приближения. Примером такого использования GANP является предлагаемый здесь гибридный алгоритм, построенный на основе алгоритма GANP, метода перебора L-классов [51, 52, 178] и эвристики лагранжевой релаксации.

Метод перебора L-классов предложен в работах А.А. Колоколова для решения задач целочисленного программирования в рамках исследований этих задач с помощью регулярных разбиений [62, 63]. С использованием регулярных разбиений введены и изучены новые классы отсечений, построены оценки числа итераций для ряда известных алгоритмов решения задач целочисленного программирования, разработаны новые алгоритмы, исследованы вопросы устойчивости задач и алгоритмов [28, 53, 62–64, 66, 67].

Работа алгоритма начинается с получения приближенного решения и нижней оценки оптимального значения целевой функции исходной ЗНП

при помощи эвристики лагранжевой релаксации. Если на основе нижней оценки выясняется оптимальность полученного решения, то работа завершается. Иначе выполняется поиск приближенного решения с помощью GANP. Подробнее эвристика лагранжевой релаксации описана ниже.

После завершения первого этапа, когда найдено некоторое допустимое решение ζ^0, начинает работу алгоритм перебора L-классов с использованием $r = (\mathbf{c}, \zeta^0)$ в качестве рекорда.

Для описания алгоритма перебора L-классов требуется ввести некоторые определения. *L-разбиение* пространства $\mathbb{R}^n$ определяется следующим образом: точки $\mathbf{x}$, $\mathbf{y}$ из $\mathbb{R}^n$ ($x \succ y$) эквивалентны, если не существует $\mathbf{z} \in \mathbb{Z}^n$ такой, что $\mathbf{x} \succeq \mathbf{z} \succeq \mathbf{y}$. Здесь $\succ, \succeq$ – символы лексикографического сравнения. Элементы данного разбиения называются *L-классами*. L-разбиение обладает свойством согласованности с лексикографическим порядком, т. е. элементы $\mathbb{R}^n/L$ могут быть лексикографически упорядочены. Пусть $V_1, V_2 \in \mathbb{R}^n/L$. Множество V_1 лексикографически больше V_2 ($V_1 \succ V_2$), если $\mathbf{x} \succ \mathbf{x}'$ для всех $\mathbf{x} \in V_1$, $\mathbf{x}' \in V_2$. Так как для ограниченного множества S фактор-множество S/L конечно, то оно может быть представлено в виде $S/L = \{V_1, V_2, \ldots, V_\theta\}$, где $V_k \succ V_{k+1}$, $k = 1, \ldots, \theta - 1$.

В процессе выполнения алгоритма перебора L-классов в порядке лексикографического возрастания просматриваются L-классы той части полиэдра ЛП-релаксации Ω, для которой значение целевой функции не превосходит величины $r - \varepsilon$. В данном случае просмотр ведется по лексикографическому возрастанию L-классов. Для получения точного решения параметр ε в случае целочисленных стоимостей полагается равным 1, а если имеются дробные стоимости, то $\varepsilon = 0$.

В [62] показано, что множество Ω в случае ЗНП обладает альтернирующей L-структурой, т.е. допустимые целочисленные точки и дробные L-классы чередуются при их лексикографическом упорядочении. В частности, $\operatorname{lexmin} \Omega \in \mathbb{Z}^n$. Поэтому перебор L-классов всегда начинается с це-

лочисленной точки. Рассмотрим схему алгоритма перебора L-классов, учитывающего специфику ЗНП [51].

Алгоритм 5.3. LCE$_{scp}$

1. Найти некоторое приближенное решение $\zeta^0 \in \mathbb{Z}^n$.

   1.1 Найти $\mathbf{x}' = \text{lexmin } \Omega$.

   1.2 Вычислить рекорд $r := \min\{(\mathbf{c}, \mathbf{x}'), (\mathbf{c}, \zeta^0)\}$.

   1.3 Положить $p := \max\{j \mid x'_j = 1, j = 1, ..., n-1\}$.

2. Найти $\varphi = \max\{j \mid x'_j = 0, j = 1, ..., p-1\}$.

   Если такого φ нет, то переход на шаг 5.

3. Начало итерации. Пусть $\mathbf{x}'' := \mathbf{x}'$. Решить задачу ЛП:

$$\mathbf{x}' = \text{lexmin}\{\mathbf{x} \in \Omega \mid (\mathbf{c}, \mathbf{x}) \leq r - \varepsilon, x_1 = x''_1, ..., x_{\varphi-1} = x''_{\varphi-1}, x_\varphi = 1\}.$$

   3.1 Если подзадача не имеет решений и $\varphi = 1$, то на шаг 5.

   3.2 Если подзадача не имеет решений и $\varphi > 1$, то

   положить $p := \varphi$ и перейти на шаг 2.

   3.3 Если $\mathbf{x}' \in \mathbb{Z}^n$ и $r > (\mathbf{c}, \mathbf{x}')$, положить $r := (\mathbf{c}, \mathbf{x}')$.

   3.4 Если $\mathbf{x}' \in \mathbb{Z}^n$, положить $p := \max\{j : x'_j = 1, j = 1, ..., n-1\}$

   и перейти на шаг 2.

4. Найти $\varphi := \min\{j : x'_j \neq \lfloor x'_j \rfloor, j = 1, ..., n\}$, перейти на шаг 3.

5. Лучшее найденное целочисленное решение ζ оптимально.

На шаге 3 при переходе к очередному L-классу часто возникают задачи ЛП, не имеющие допустимых решений. Чтобы избежать ненужных временных затрат при решении этих задач в гибридном алгоритме используется метод группового тестирования, предложенный Л.А. Заозерской в [51, 52]. Процедура группового тестирования позволяет анализировать не один очередной L-класс, а сразу несколько. Для этого на шаге 3 в случае отсутствия L-класса, где $x_j = x''_j$ для $j = 1, \ldots, \varphi - 1$, и $x_\varphi = 1$,

решается подзадача поиска вектора $\mathbf{x}'$, равного

$$\text{lexmin}\{\mathbf{x} \in \Omega \mid (\mathbf{c}, \mathbf{x}) \leq r - \varepsilon, x_1 = x''_1, ..., x_{j_0-1} = x''_{j_0-1}, \sum_{j=j_0}^{j_0+n_0-1} x_j \geq 1\}, \quad (5.6)$$

где n_0 – число нулей, расположенных подряд в векторе $\mathbf{x}''$ перед первой дробной координатой, j_0 – индекс первого нуля в этой группе.

Если во множестве $\{\mathbf{x} \in \Omega \mid (\mathbf{c}, \mathbf{x}) \leq r - \varepsilon\}$ не существует L-классов, для которых $x_j = x''_j,\ j = 1, ..., j_0 - 1$, лексикографически больших $\mathbf{x}''$, то задача (5.6) не имеет решения. Иначе будет найден очередной L-класс и процесс перебора продолжится с него.

До решения задачи (5.6) имеет смысл провести ее проверку на существование допустимого решения. Рассмотрим следующую вспомогательную подзадачу ЛП: найти

$$\min \sum_{j=j_0}^{n} c_j x_j \quad (5.7)$$

при условиях

$$\mathbf{x} \in \Omega, \quad (5.8)$$

$$\sum_{j=j_0}^{j_0+n_0-1} x_j \geq 1, \quad (5.9)$$

$$x_j = x''_j, j = 1, ..., j_0 - 1. \quad (5.10)$$

Если оптимальное решение задачи (5.7) – (5.10) больше, чем $r' = r - \sum_{j=1}^{j_0-1} c_j x''_j - \varepsilon$, то задача поиска лексикографического минимума (5.6) не имеет решения. Для получения нижней оценки оптимума задачи (5.7) – (5.10) можно воспользоваться, например, приближенным решением двойственной задачи, полученным с помощью жадного алгоритма (см. [52]). Данная эвристика имеет значительно меньшую трудоемкость по сравнению с лексикографическим двойственным симплекс-методом, применяемым при решении задачи (5.6).

На этапе тестирования в предлагаемом гибридном алгоритме помимо жадного алгоритма используется эвристика лагранжевой релаксации.

В данной эвристике осуществляется релаксация всех линейных ограничений, определяемых неравенствами (5.2), и для каждого из них вводится в рассмотрение множитель Лагранжа. Пересчет множителей Лагранжа выполняется в соответствии с алгоритмом субградиентной оптимизации [141]. С периодичностью в γ итераций по имеющимся множителям строится двойственно допустимое решение $\mathbf{y}$ с использованием процедуры DualFeas, предложенной в [119]. Длина периода γ подбирается экспериментально и фиксирована от начала работы алгоритма до конца. На основе решения $\mathbf{y}$ посредством эвристики редуцированных стоимостей [119] получается приближенное решение $\tilde{\zeta}$ вспомогательной задачи наименьшего покрытия множества (5.7) – (5.10) при условии $\mathbf{x} \in \{0,1\}^n$. Если $(\mathbf{c}, \tilde{\zeta}) < r$, то полагаем $r := (\mathbf{c}, \tilde{\zeta})$. Пересчет множителей Лагранжа завершается, когда в течение τ итераций эвристики не происходит улучшения нижней оценки оптимума задачи (5.7) – (5.10) или если на очередной итерации выясняется, что значение этой оценки превосходит $r' - \varepsilon$.

Жадный алгоритм и эвристика лагранжевой релаксации в гибридной схеме применяются не только в процедуре группового тестирования, но и перед каждым решением вспомогательной задачи ЛП на шаге 3. В случае, если удается установить неразрешимость очередной подзадачи, поиск лексикографического минимума не выполняется.

Описанный гибридный алгоритм за конечное число шагов находит решение ЗНП. При этом количество просматриваемых L-классов, как правило, оказывается существенно меньше, чем в случае непосредственного использования алгоритма перебора L-классов. Гибридный алгоритм предложен автором совместно с А.А. Колоколовым и Л.А. Заозерской.

5.1.3. Экспериментальные результаты

Тестовые задачи и экспериментальные исследования

В настоящее время имеется достаточно большой объем тестовых задач для экспериментального исследования и сравнения алгоритмов решения ЗНП. Множество тестовых задач можно разбить на три основные группы: задачи специальной структуры [34, 91, 187, 205], прикладные задачи [119, 141] и задачи, построенные с использованием датчика псевдослучайных чисел [111, 120, 128]. Значительная часть этих задач доступна в сети Интернет и используется многими авторами. В частности, большое количество примеров содержится в электронной библиотеке OR-Library [126] (http://people.brunel.ac.uk/m̃astjjb/jeb/info.html).

При сравнении алгоритмов многими авторами используются невзвешенные ЗНП, возникающие в комбинаторике. Большой интерес представляют задачи на системах троек Штейнера (серия **Stein**). Как было отмечено в [187], задачи этого типа представляют большую трудность для решения с помощью точных алгоритмов. Судя по публикациям оптимальные решения известны для таких задач с размерностью до 243 переменных [239, 256]. В экспериментах также используются примеры серии **CLR**, связанные с известной задачей Эрдеша [105, 172] о раскраске гиперграфа.

Для сравнения алгоритмов решения ЗНП в настоящее время используется большое число задач, построенных с помощью датчиков псевдослучайных чисел. Наиболее известны серии **4,5,6**, предложенные в работе Э. Балаша и А. Ху [120], а также серии задач большей размерности **A,B,...,H**, сгенерированные Дж. Бисли (см., например, [125]) аналогично сериям **4,5,6**. Эти задачи построены с заданной разреженностью матрицы (2%, 5%, 10% и 20% единиц), причем каждая строка содержит по крайней мере две единицы и каждый столбец – по крайней мере одну. Веса выбраны случайным образом в диапазоне от 1 до 100. Широкий разброс весов и

небольшая плотность этих задач позволяют существенно уменьшить число анализируемых вариантов в точных алгоритмах по сравнению с полным перебором (см., например, [128]). Исходные данные описанных задач доступны в библиотеке OR-Library. Достаточно трудные случайные задачи размерности $n = 100, m = 100$ построены в [111] (серия **E5**). В этих задачах все веса равны единице, а матрица содержит от 4 до 5 единиц в каждой строке. Информация об этих и других тестовых примерах для ЗНП может быть найдена в обзоре [40].

Описанный в п. 5.1.2 гибридный алгоритм и эвристика GANP были запрограммированы на языке Borland Pascal 7.0. В экспериментах использовались серии тестовых задач **4-6, A-H**, **CLR** и **Stein** из электронной библиотеки OR-Library [126]. Кроме того, при тестировании некоторых алгоритмов использовались задачи серии **E5**.

Выбор параметров генетического алгоритма GANP

С целью снижения вычислительных затрат до начала работы ГА размерность исходной задачи понижается путем исключения из матрицы **Q** тех столбцов, которые заведомо не могут войти ни в одно решение, генерируемое ГА при данном параметре α. Для этого из исходной задачи выделяется «ядро», состоящее из столбцов с индексами из $\cup_{i=1}^{m} U_i(\alpha)$.

В проведенных экспериментах при параметре $\alpha = 10$ за 10 запусков ГА нашел оптимум хотя бы один раз для всех задач серий **4-6, A-D,F**, а для всех задач из **E,G** и **H** кроме **H.1** и **H.2** были найдены наилучшие известные решения. Уменьшение параметра α приводит к задачам меньшей размерности, которые, как правило, решаются быстрее и с большей точностью, однако оптимальное решение исходной ЗНП в некоторых тестовых задачах при этом бывает потеряно. Например при $\alpha = 5$, оптимальное решение задачи **5.3** после выделения «ядра» становится недопустимым. С другой стороны, при $\alpha = 5$ удается найти наилучшее известное решение

Таблица 5.1. Параметры задач

Серия	m	n	n'	$\varrho(\%)$	S	χ
4	200	1000	640-696	2	10	0.97
5	200	2000	628-672	2	10	0.89
6	200	1000	293-315	5	5	0.70
A	300	3000	694-724	2	5	0.86
B	300	3000	312-329	5	5	0.58
C	400	4000	720-737	2	5	0.86
D	400	4000	328-341	5	5	0.65
E	500	5000	171-180	10	5	0.34
F	500	5000	91-96	20	5	0.14
G	1000	10000	808-817	2	5	0.81
H	1000	10000	353-370	5	5	0.48

для задачи **H.2**, а среднее время достижения оптимума на серии **4** уменьшается более чем в 3 раза. При решении задач с единичными стоимостями редукция не применяется и параметр α полагается равным n.

В Таблице 5.1 приведены исходные размерности задач, число столбцов n' после редукции с параметром $\alpha = 10$, а также процент единиц ϱ в матрицах тестовых задач. В столбце S указано число задач в серии, а в столбце χ – доля успешно решенных подзадач линейной релаксации при выполнении ЛП-кроссинговера. Тестирование алгоритма GANP выполнялись на ПК Pentium с тактовой частотой процессора 100 МГц и 16 Мб ОЗУ.

С целью настройки параметров ГА в [34] были проведены эксперименты при различных значениях вероятности мутации P_{m} и размера популяции N. Для задач со случайными данными была выбрана вероятность мутации, равная 0.1. При решении задач серии **CLR** алгоритм GANP оказался весьма чувствителен к параметру P_{m}, в связи с этим P_{m} подбиралась экспериментально для каждой задачи этой серии. Размер популяции был выбран равным 100 для всех рассматриваемых далее задач.

Сравнение GANP с другими эвристиками при решении задач, построенных случайным образом

В Таблицах 5.2 и 5.3 алгоритм GANP сравнивается с генетическим алгоритмом Дж. Бисли и П. Чу [127] (обозначен BC) и алгоритмом лагранжевой релаксации А. Капрары, М. Фишетти и П. Тота [141] (обозначен CFT). В Таблице 5.2 также приведены результаты модификации алгоритма GANP, где вместо ослабленной ЗОР в операторе ЛП-кроссинговера решается ЗОР, соответствующая определению 3.1 (столбец обозначен GAOR). В экспериментах проводилось по 10 испытаний GANP и GAOR для каждой задачи при параметрах $N = 100$, $P_{\rm m} = 0.1$, $t_{max} = 10000$. Таблицы содержат следующие величины.

Обозначим через $P_{\rm bst}$ частоту получения наилучшего из найденных ГА решений при 10 испытаниях. Столбец σ содержит средний процент отклонения от оптимума или наилучшего известного решения, т. е. $\sigma = \sum_{k=1}^{10} \frac{F_k - F^*}{10F^*} \cdot 100\%$, где F_k – стоимость решения, найденного при k-м испытании, а F^* – стоимость оптимального или наилучшего известного решения. Обозначим через T_{sol} среднее время последнего обновления решения; T_{ex} – среднее полное время работы алгоритма при 10000 итераций (в секундах ПК Pentium-100). Время счета алгоритма BC приближенно оценено на основании данных, приведенных в [127] с использованием таблицы быстродействия ЭВМ [168]. В Таблице 5.2 указанные величины усреднены по всем задачам для каждой серии **4-6** и **A-H**.

Полное время счета алгоритма CFT [141] на всех задачах было одинаковым, и согласно таблице [168] соответствует 2112 секундам Pentium-100). Трудоемкость одной итерации GAOR близка к трудоемкости одной итерации GANP, поэтому в Таблице 5.2 величины T_{sol} и T_{ex} для GAOR не приводятся.

В соответствии с данными из [127] при 10 запусках алгоритм BC находил хотя бы раз оптимальное решение для всех задач **4-6** и **A-D** кроме

Таблица 5.2. Эксперименты с сериями **4-6** и **A-D**

Серия	CFT	BC				GANP				GAOR	
	T_{sol}	P_{bst}	σ	T_{sol}	T_{ex}	P_{bst}	σ	T_{sol}	T_{ex}	P_{bst}	σ
4	2.9	0.88	0.07	45.0	315	0.86	0.09	33.5	369	0.71	0.15
5	1.4	0.76	0.17	27.2	552	0.7	0.27	37.7	318	0.68	0.3
6	4.1	0.94	0.07	25.2	319	0.86	0.24	41.0	486	0.84	0.26
A	47.2	0.86	0.20	66.0	788	0.44	0.35	71.8	1693	0.68	0.23
B	3.3	1.00	0.00	68.6	938	1.00	0.00	20.8	344	1.0	0.0
C	29.2	0.68	0.41	87.9	1141	0.74	0.26	52.4	721	0.82	0.1
D	7.6	0.96	0.06	101.7	1373	0.94	0.08	23.3	583	1.0	0.0
E	54.6	0.74	0.92	124.6	3088	1.0	0.0	26.4	1195	1.0	0.0
F	48.2	0.86	1.08	69.2	5470	0.86	1.08	47.2	3798	0.82	1.39
G	223.4	0.24	1.16	366.1	3440	0.48	0.54	196.9	888	0.64	0.55
H	379.8	0.44	1.04	587.1	5596	0.6	0.64	172.4	4270	0.4	0.64

задачи **5.3**. Алгоритм CFT для каждой задачи запускался однократно, при этом во всех 45 случаях был найден оптимум. Результаты этих экспериментов изложены в работе [141].

Из сравнения столбцов GANP и GAOR в Таблице 5.2 можно сделать вывод о том, что существенного различия между двумя рассмотренными вариантами ЛП-кроссинговера не наблюдается. В связи с этим далее приводятся только результаты GANP, где в ЛП-кроссинговере решается ослабленная ЗОР.

Как видно из Таблицы 5.1, в большинстве рассмотренных серий частота успешного выполнения ЛП-кроссинговера χ составляла не менее 0.5. Для исследования эффекта применения данного оператора были проведены дополнительные эксперименты с задачей **4.1**, в которых подсчитывалась частота получения оптимума ЗНП при использовании ЛП-кроссинговера и без него. После 1000 итераций алгоритма эти частоты составили соответственно 0.6 и 0.0, а после 10000 итераций они были равны 0.87 и 0.66. При решении других задач этой серии использование ЛП-кроссинговера также приводило к существенному увеличению качества по-

лучаемых решений.

В Таблице 5.3 приведена более подробная информация по экспериментам с задачами серий **E-H**. Здесь F' – стоимость наилучшего известного решения из литературы [141, 295]; LP – оценка оптимума снизу, полученная решением линейной релаксации ЗНП с последующим округлением вверх; F – наилучшее значение целевой функции, найденное эвристикой. В случае, если это значение совпадает со стоимостью лучшего известного решения, оно выделяется жирным шрифтом. Полное время работы T_{ex} алгоритма BCH на сериях **E,F,G** и **H** в среднем составляет, соответственно, 3089.2, 5470, 3428.4, и 5596.4 секунд ПК Pentium-100. Полное время работы T_{ex} алгоритма CFT и на этих сериях приближенно соответствует 2212 секундам ПК Pentium-100. Если на основе нижних оценок из [141,295] можно утверждать, что наилучшие известные решения из литературы являются оптимальными, эти решения указываются с символом '*' в столбце F'.

На основании проведенного вычислительного эксперимента можно отметить, что при указанных параметрах генетические алгоритмы GANP и BC показали близкие результаты по качеству полученных решений. При этом на сериях малой размерности **4-6** и **A-D** генетические алгоритмы имеют бóльшее время поиска оптимальных решений по сравнению с CFT, а на сериях **E-H** среднее время последнего обновления рекорда в GANP меньше, чем у других алгоритмов.

На серии **E** алгоритм GANP превосходит два других алгоритма; на части задач серий **F** и **H** он уступает CFT по времени решения или по качеству найденных решений, а на части задач более эффективен по сравнению с ним. В целом по качеству решений GANP уступает алгоритму CFT только на двух из 65 рассмотренных случайных задач.

Алгоритм GANP опубликован в [173]. Позднее в работе [295] М. Ягиура, М. Кишида и Т. Ибараки (M. Yagiura, M. Kishida, T. Ibaraki) предложили гибридный алгоритм, сочетающий метод лагранжевой релаксации

Таблица 5.3. Эксперименты с сериями **E-H**

			CFT		BC				GANP				
	F'	LP	F	T_{sol}	F	P_{bst}	σ	T_{sol}	F	P_{bst}	σ	T_{sol}	T_{ex}
E.1	29*	22	**29**	12	**29**	1.0	0.0	17	**29**	1.0	0.0	1	562
E.2	30	23	**30**	181	**30**	0.4	2.0	266	**30**	1.0	0.0	95	1768
E.3	27*	21	**27**	42	**27**	0.3	2.6	85	**27**	1.0	0.0	23	1345
E.4	28*	22	**28**	12	**28**	1.0	0.0	238	**28**	1.0	0.0	11.0	1234
E.5	28*	22	**28**	16	**28**	1.0	0.0	16	**28**	1.0	0.0	2	1067
F.1	14*	9	**14**	15	**14**	1.0	0.0	34	**14**	1.0	0.0	8	1699
F.2	15*	10	**15**	14	**15**	1.0	0.0	34	**15**	1.0	0.0	1	1549
F.3	14*	10	**14**	110	**14**	1.0	0.0	117	**14**	1.0	0.0	55	1765
F.4	14*	9	**14**	14	**14**	1.0	0.0	92	**14**	1.0	0.0	20	5686
F.5	13*	8	**13**	89	**13**	0.3	5.4	67	**13**	0.3	5.4	151	8290
G.1	176	160	**176**	65	**176**	0.2	1.0	451	**176**	0.7	0.3	96	795
G.2	154	143	**154**	346	155	0.5	1.5	159	**154**	0.5	0.7	227	837
G.3	166	149	**166**	432	**166**	0.1	1.1	312	**166**	0.1	0.8	319	973
G.4	168	149	**168**	167	**168**	0.2	1.4	665	**168**	0.4	0.7	173	885
G.5	168	149	**168**	105	**168**	0.2	0.8	242	**168**	0.7	0.2	170	952
H.1	63	49	**63**	642	64	1.0	1.6	743	64	1.0	1.6	91	1574
H.2	63	49	**63**	392	64	1.0	1.6	234	64	1.0	1.6	35	9650
H.3	59	46	**59**	690	**59**	0.9	0.2	796	**59**	1.0	0.0	493	5620
H.4	58	44	**58**	105	**58**	0.4	1.6	962	**58**	1.0	0.0	218	3458
H.5	55	43	**55**	69	**55**	0.9	0.2	198	**55**	1.0	0.0	25	1049

и локальный поиск. По приближенным оценкам на сериях **4-6, A-H** алгоритм [295] обладает бóльшим быстродействием, чем все рассмотренные выше алгоритмы. К сожалению, у нас нет информации о том, применялись ли алгоритмы BC, CFT и алгоритм [295] к задачам специальной структуры, которые будут рассмотрены ниже.

Эксперимент с задачами, имеющими комбинаторную постановку

Примеры серии **CLR** связаны с известной задачей П. Эрдеша о раскраске гиперграфа [172]: найти значение $m_d(r)$, равное минимальному числу сочетаний по r элементов, которые достаточно выбрать из базового d-элементного множества V, чтобы при любой раскраске в 2 цвета элементов V, хотя бы одно из выбранных r-элементных сочетаний оказывалось монохроматическим. Предполагается, что $d \geq 2r - 1$.

При фиксированных r и d эта задача может быть сведена к ЗНП (см., например, [205]), в которой M – множество всевозможных раскрасок базового множества, U – множество всех сочетаний по r элементов из V. Полагаем, что сочетание покрывает те раскраски, при которых оно монохроматично. С учетом симметрии половину раскрасок можно не рассматривать. Тогда ЗНП имеет размерность $n = \binom{d}{r}, m = 2^{d-1} - 1$, и $m_d(r)$ совпадает с мощностью оптимального покрытия. Здесь рассматривается случай $r = 4$. Даже для этого случая оптимальные решения задачи не известны. В Таблице 5.4 приведены размерности получаемых ЗНП, наилучшие известные из литературы [201, 205, 254] верхние оценки оптимума (столбец F'), а также число элементов покрытий, найденных с помощью GANP. Номера задач в серии **CLR** соответствуют значениям параметра d. Столбцы P_{bst} и T_{sol} имеют тот же смысл, что и в Таблице 5.3. Как и в экспериментах, описанных выше, $N = 100$, $t_{max} = 10000$. Частота срабатывания ЛП-кроссинговера χ и вероятность мутации P_{m} при решении этих задач также приведены в Таблице 5.4.

Таблица 5.4. Эксперименты с сериями **CLR** и **Stein**

Задача	m	n	F'	GANP	P_{bst}	T_{sol}	P_{m}	χ
CLR.9	255	126	26	**26**	1.0	15.32	0.05	0.8
CLR.10	511	210	25	**25**	1.0	27.0	0.05	0.6
CLR.11	1023	330	23	**23**	1.0	274.6	0.05	0.6
CLR.12	2047	495	26	*23*	0.2	979.7	0.02	0.7
CLR.13	4095	715	26	*23*	0.3	4615.4	0.01	0.6
Stein.27	117	27	18*	**18**	1	1.2	0.1	0.01
Stein.45	330	45	30*	**30**	1	12.6	0.1	0.03
Stein.81	1080	81	61*	**61**	1	16.6	0.02	0.1
Stein.135	3015	135	103*	**104**	0.2	212.4	0.01	0.1
Stein.243	9801	243	198*	*198*	0.6	536.1	0.005	0.4

Для случаев $d = 12$ и $d = 13$ известные оценки улучшены с помощью GANP, а для $d = 9, 10, 11$ найдены решения с такими же значениями целевой функции, что и известные ранее (см., например, [205, 254]).

Заметим, что решения из 23 покрывающих элементов могут быть получены для произвольного $d > 11$ и без использования ЭВМ. Достаточно заметить, что если набор сочетаний $\{S_1, ..., S_k\}$ (где $|S_i| = 4, i = 1, ..., k$) удовлетворяет условиям задачи при $d = d'$, то этот же набор удовлетворяет условиям и при любом $d > d'$.

Таблица 5.4 содержит также результаты работы GANP на задачах серии **Stein**. Система троек Штейнера состоит из $m = \frac{n(n-1)}{6}$ трехэлементных подмножеств базового n-элементного множества, где $n \equiv 1, 3(mod\ 6)$, и любая пара элементов базового множества содержится ровно в одной тройке Штейнера. Задача состоит в поиске наименьшего покрытия всех троек элементами базового множества (предполагается, что каждый элемент покрывает все тройки системы, содержащие его).

В экспериментах с этими примерами известные из [239, 256] оптимальные решения найдены GANP во всех задачах, кроме **Stein.135**. Покрытия мощностью 103 и 198 для задач **Stein.135** и **Stein.243** опубликованы в [253] примерно в то же время, когда был предложен GANP.

Влияние кроссинговера на результаты генетического алгоритма

Для исследования влияния кроссинговера на результаты работы GANP были проведены эксперименты на задачах из разных классов: **4.1, G.2, Stein.243** и **CLR.12**. При этом рассматривался GANP с ЛП-кроссинговером, с равномерным кроссинговером и без кроссинговера.

Предварительно для настройки параметра вероятности применения равномерного кроссинговера P_c было выполнено по 10 запусков алгоритма GANP со значениями $P_c = 0, 0.1, \ldots, 1$. Прочие параметры ГА были теми же, что указаны выше. Среднее значение частоты получения оптимума на задачах **4.1, G.2** и **Stein.243** оказалось максимальным при $P_c = 0.8$, поэтому в последующих экспериментах с равномерным кроссинговером рассматривалось только значение P_c, равное 0.8.

Поскольку размер популяции N существенно влияет на разнообразие особей в X^t, данный параметр может влиять и на эффективность кроссинговера. Ввиду этого для каждого варианта ГА было выполнено по 10 запусков с разными значениями N. Прочие параметры имели те же значения, что и ранее. Результаты экспериментов приведены на Рисунках 5.1 – 5.4. Здесь «P* без кросс.» – частота получения оптимума без использования кроссинговера, «P* ЛП-кросс.» – частота получения оптимума при использовании ЛП-кроссинговера и «P* униф. кросс., Pc=0.8» – частота получения оптимума при использовании равномерного кроссинговера.

Как видно из Рисунков 5.1 и 5.2, в задачах **4.1** и **G.2** ГА с ЛП-кроссинговером, как правило, имеет преимущество перед двумя другими вариантами алгоритма при малом размере популяции, и это преимущество усиливается с увеличением N. Дополнительные эксперименты показали, что такая тенденция типична для большинства задач разреженных серий **4-6** и **A-D,G,H**. Заметим, что разрыв двойственности в этих задачах относительно мал (см., например, [128] и Таблицу 5.3), и можно предположить, что эффективность ЛП-кроссинговера здесь вызвана близостью оп-

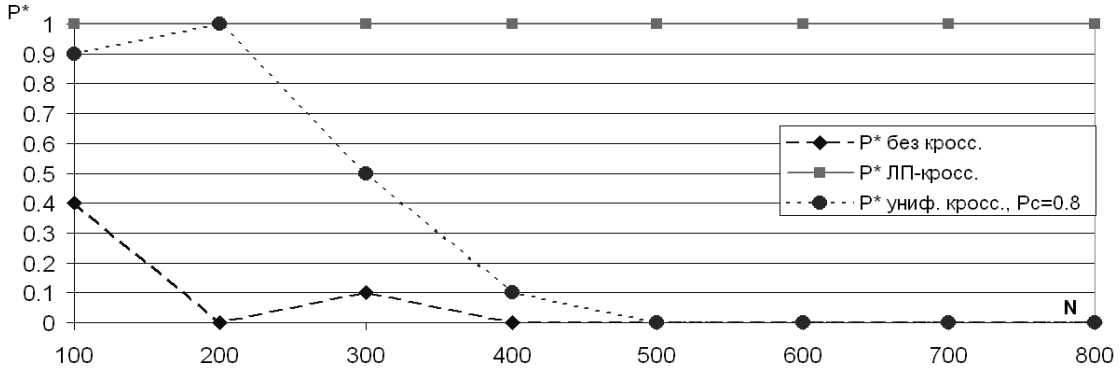

Рис. 5.1. Частота получения оптимума в задаче **4.1**.

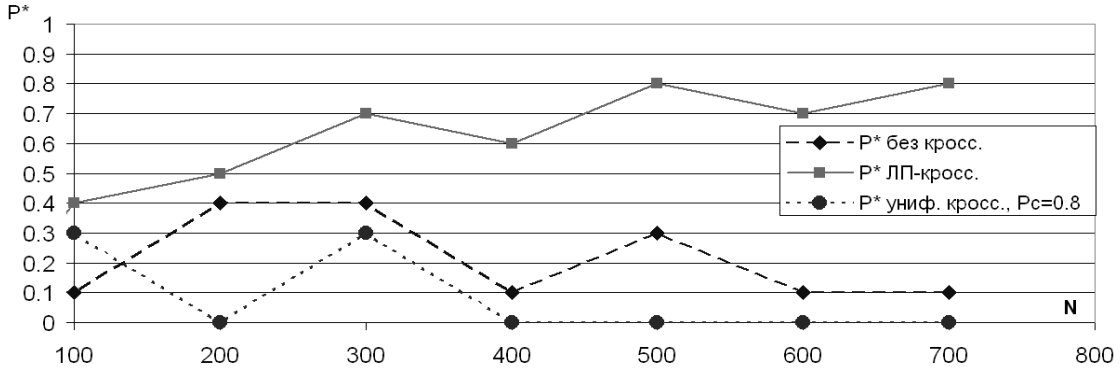

Рис. 5.2. Частота получения оптимума в задаче **G.2**.

тимума ЗНП к решению ЛП-релаксации. Высокая частота срабатывания ЛП-кроссинговера на задачах со случайными данными хорошо согласуется с исследованиями асимптотических свойств таких задач, проведенными К. Верцеллисом [287]. Из результатов К. Верцеллиса следует, что совпадение значений целевой функции оптимальных решений ЗНП и ее ЛП-релаксации является «типичной» ситуацией в достаточно широком диапазоне значений разреженности матрицы. Причина высокой частоты получения *целочисленного* решения ЛП-релаксации с помощью симплекс-метода в операторе ЛП-кроссинговера остается невыясненной.

Результаты экспериментов на задачах серий **Stein** и **CLR** существен-

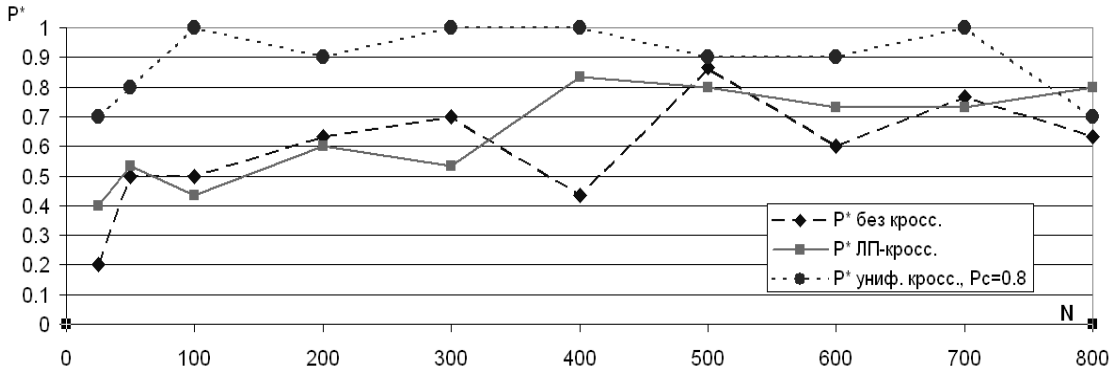

Рис. 5.3. Частота получения оптимума в задаче **Stein.243**.

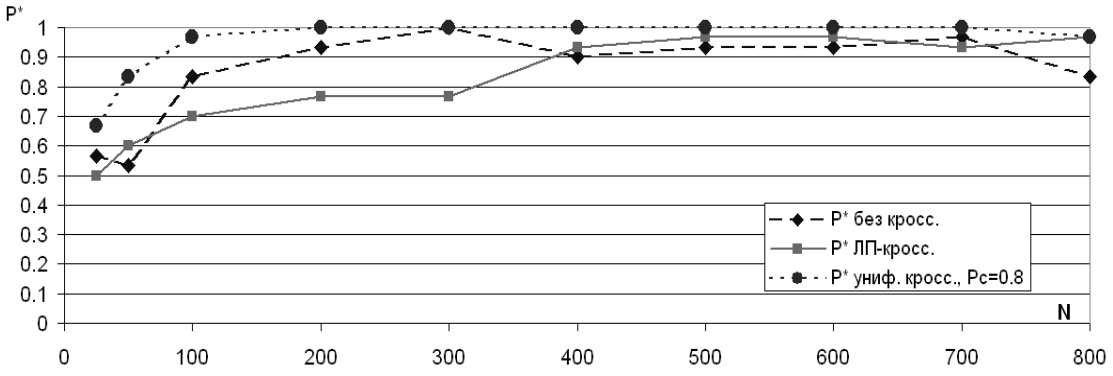

Рис. 5.4. Частота получения оптимума в задаче **CLR.12**.

но отличаются от рассмотренных выше. Как видно из Рисунка 5.3 и Рисунка 5.4, в задачах **Stein.243** и **CLR.12** графики ГА с ЛП-кроссинговером близки к графикам ГА без кроссинговера. Наибольшая частота получения оптимума при различных размерах популяции на этих задачах достигается ГА с равномерным кроссинговером. Аналогичная тенденция наблюдается и на других задачах серий **Stein** и **CLR**. По-видимому, причина низкой эффективности ЛП-кроссинговера на этих сериях состоит в большом расстоянии между оптимальным решением ЗНП и решением ЛП-релаксации в задачах с такой комбинаторной структурой: разрыв двойственности на серии CLR составляет от 16% до 37% [205], а в задачах серии **Stein** значение

целевой функции оптимального решения ЛП-релаксации составляет $n/3$ (см., например, [187]), что в сравнении с данными Таблицы 5.4 дает разрыв двойственности в 50% и более.

Таким образом, результаты экспериментов свидетельствуют о том, что применение ЛП-кроссинговера в GANP целесообразно при решении задач, где разрыв двойственности невелик. В противном случае предпочтительнее использовать равномерный кроссинговер.

Экспериментальное исследование гибридного алгоритма

Эксперименты с гибридным алгоритмом выполнялись на ПК Pentium-II с объемом ОЗУ 32 Мб и процессором Celeron, работающим на тактовой частоте 460 МГц. Время счета тестируемых программ на данном ПК приблизительно в 2.5 – 3 раза меньше по сравнению с ПК Pentium-100.

Для исследования гибридного алгоритма использовались тестовые задачи серий **4** и **6** из электронной библиотеки OR-Library [126], а также серия **E5** из [111], предложенная Ю.А. Кочетовым.

Настраиваемые параметры ГА, входящего в гибридный алгоритм, были выбраны следующим образом: $N = 100$, $P_{\mathrm{m}} = 0.2$, $t_{max} = 1000$, $\alpha = 10$. На начальном этапе генетический алгоритм выполнялся дважды, а эвристика лагранжевой релаксации – один раз. Параметр τ, используемый в критерии остановки метода лагранжевой релаксации, положен равным 500. Построение допустимого покрытия подзадачи в этой эвристике выполнялось через каждые $\gamma = 10$ пересчетов множителей Лагранжа. В процессе перебора L-классов тестирование подзадач ЛП осуществлялось при числе переменных в подзадаче не менее 30.

В таблицах, описывающих эти эксперименты, столбец LP содержит оптимальное значение целевой функции ЛП-релаксации; столбцы T_{opt} и T_{ex} содержат время обнаружения оптимума и полное время счета программы; F^* обозначает оптимальное значение целевой функции. В столбце «проц.»

Таблица 5.5. Решение задач серии **4**

задача	F^*	LP	LCE_{SCP}				гибридный алгоритм				
			T_{opt}	T_{ex}	L_0	L_1	T_{opt}	T_{ex}	проц.	L_0	L_1
4.1	429	429.0	96	365	18	0	3	3	LR	0	0
4.2	512	512.0	594	1461	86	0	2	2	LR	0	0
4.3	516	516.0	87	362	30	0	2	2	LR	0	0
4.4	494	494.0	318	947	59	0	16	16	GA	0	0
4.5	512	512.0	97	352	29	0	2	2	LR	0	0
4.6	560	557.3	149	842	49	0	23	57	GA	0	0
4.7	430	430.0	188	400	24	0	2	2	LR	0	0
4.8	492	488.7	164	764	31	1	16	79	GA	0	0
4.9	641	638.5	511	1485	123	3	100	128	LR	0	2
4.10	514	513.5	304	537	77	0	2	2	LR	0	0

указана процедура, с помощью которой обнаружен оптимум (LCE – перебор L-классов; Grd – жадная эвристика подъема; GA – генетический алгоритм GANP; LR – эвристика лагранжевой релаксации до перебора L-классов). Через L_0 и L_1 здесь обозначено число L-классов, просмотренных до обнаружения оптимума и после него.

В Таблице 5.5 приведены результаты экспериментов с задачами серии **4**. Как видно из таблицы, в большинстве случаев оптимальное решение было найдено уже методом лагранжевой релаксации и нижняя оценка оптимума целевой функции, полученная этим алгоритмом, сразу показывала оптимальность обнаруженного решения. Перебор L-классов потребовался только для одной из десяти задач. Благодаря использованию метода лагранжевой релаксации общее время вычислений гибридного алгоритма на этой серии в среднем оказалось примерно в 26 раз меньше, чем время работы алгоритма LCE_{SCP}. Такое ускорение алгоритма вызвано в первую очередь малой величиной разрыва двойственности.

Серии **6** и **E5** содержат задачи с бóльшим разрывом между решениями ЗНП и ЛП-релаксации. В связи с этим трудоемкость решения задач **6.1** – **6.5** алгоритмами, использующими двойственные оценки, как правило, выше по сравнению с задачами **4.1** – **4.10** (см., например, [119, 185]).

Таблица 5.6. Решение задач серий **6** и **E5**

задача	F^*	LP	LCE_{SCP}				гибридный алгоритм				
			T_{opt}	T_{ex}	L_0	L_1	T_{opt}	T_{ex}	Proc.	L_0	L_1
6.1	138	133.1	761	1354	51	1	33	288	GA	0	10
6.2	146	140.4	130	751	39	8	45	199	GA	0	14
6.3	145	140.1	183	543	22	1	26	128	LR	0	3
6.4	131	129.0	61	531	18	1	22	88	LR	0	2
6.5	161	153.4	1322	2120	184	3	29	304	GA	0	15
E5.1	20	17.4	279	326	104	10	11	58	GA	0	11
E5.2	19	17.7	248	253	74	0	6	15	GA	0	1
E5.3	20	17.3	431	448	211	5	8	67	GA	0	16
E5.4	19	16.6	114	123	43	3	6	27	GA	0	6
E5.5	20	17.7	0	56	0	8	0	46	Grd	0	8
E5.6	19	16.9	268	276	89	1	87	91	LCE	25	1
E5.7	20	17.6	609	645	170	6	10	75	GA	0	15
E5.8	21	17.9	174	323	38	33	26	148	GA	0	35
E5.9	19	16.9	77	107	29	5	7	32	GA	0	6
E5.10	20	17.5	117	171	26	14	8	67	GA	0	16

В экспериментах при решении серий **6** и **E5** гибридным алгоритмом большую часть времени занимал просмотр дробных L-классов уже после обнаружения оптимального решения. Во всех задачах серий **6** и **E5**, кроме задачи **E5.6**, оптимальное решение было найдено эвристиками еще до начала перебора L-классов. В результате общее число просматриваемых L-классов в задачах серии **6** сократилось в среднем примерно в 7 раз, а в задачах серии **E5** – более чем в 6 раз. Предложенный гибридный алгоритм затрачивал на решение задач серии **6** в среднем почти в 5 раз меньше времени по сравнению с LCE_{SCP}, а для серии **E5** ускорение составило 4.5 раза.

Заключение

Применение оператора ЛП-кроссинговера в GANP целесообразно при решении задач, где разрыв двойственности невелик. В противном случае предпочтительнее использовать равномерный кроссинговер.

Остается открытым вопрос теоретического обоснования высокой вероятности получения симплекс-методом оптимальных решений подзадач

ЗНП, возникающих в ЛП-кроссинговере (см. пп. 5.1.3 и 5.1.3).

При решении задач библиотеки OR-Library предложенный ГА показал результаты, не уступающие результатам ГА с двоичным представлением Дж. Бисли и П. Чу [127]. В частности, для всех 50 тестовых задач со случайными данными при размерности до 5000 переменных этим алгоритмом найдены оптимальные решения. Кроме того, для двух индивидуальных задач, связанных с вопросом П. Эрдеша о раскраске гиперграфа, получены решения, улучшающие известные из литературы оценки оптимума. Разработанный ГА может применяться как метод нахождения начального приближения для точных алгоритмов, а также для быстрого поиска приближенных решений задач большой размерности.

Эксперименты с предложенным гибридным алгоритмом показали перспективность используемого подхода. Применение ГА и эвристики лагранжевой релаксации в этом алгоритме позволило существенно сократить время поиска оптимума и полное время работы алгоритма.

5.2. Задача управления поставками продукции

В данном разделе рассматривается *задача управления поставками продукции*, состоящая в минимизации стоимости доставки продукции от множества поставщиков до множества потребителей. При этом допустимый размер каждой открытой поставки ограничен снизу и сверху, размер потребления для каждого потребителя ограничен снизу, а функции стоимости поставки линейны при ненулевых объемах поставки. Для рассматриваемой задачи предлагаются два гибридных генетических алгоритма и проводится их исследование и сравнение. Также здесь проводится анализ сложности приближенного решения задачи управления поставками. С одной стороны, в п. 5.2.1 показано, что в общем случае отыскание приближенного решения с какой-либо приемлемой оценкой точности не представляется возможным. С другой стороны, для случая одного потребителя в п. 5.2.2 предложен

эффективный 2-приближенный жадный алгоритм, используемый далее в одном из ГА, и, более того в этом частном случае существует вполне полиномиальная аппроксимационная схема (см. [41, 44, 144, 250]).

Первый из предложенных ГА основывается на использовании жадного декодера, при этом допустимое решение формируется в результате многократного применения жадного 2-приближенного алгоритма (п. 5.2.3). Вместо жадного алгоритма в данном ГА может быть использован алгоритм одной из известных аппроксимационных схем. Во втором предложенном ГА используется двоичная кодировка решений и новый оператор ЦЛП-кроссинговера, в котором решается ослабленная задача оптимальной рекомбинации (п. 5.2.4). Проведенные в п. 5.2.5 вычислительные эксперименты показывают, что ГА с ЦЛП-кроссинговером имеет преимущество по качеству получаемых решений в сравнении с первым ГА и в сравнении с пакетом решения задач ЦЛП CPLEX. В то же время ГА с использованием жадного декодера оказался наиболее предпочтительным с точки зрения возможности получения допустимого решения за приемлемое время.

Оба предложенных ГА основываются на стационарной стратегии управления популяцией. Выбор родительских особей осуществляется с помощью турнирной селекции. В данном разделе мы будем предполагать, что при выполнении условия остановки ГА начинают работу заново. Процесс вычислений продолжается до заданного ограничения на время вычислений T. Лучшее из найденных решений за время T рассматривается как окончательный результат работы алгоритма.

Задача управления поставками имеет следующую постановку.

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} z_{ij} a_{ij} + c_{ij} x_{ij}, \tag{5.11}$$

$$\sum_{i=1}^{m} x_{ij} \geq A_j, \ j = 1, ..., n, \tag{5.12}$$

$$\sum_{j=1}^{n} x_{ij} \le M_i,\ i = 1, ..., m, \tag{5.13}$$

$$z_{ij} \in \{0, 1\},\ i = 1, ..., m,\ j = 1, ..., n, \tag{5.14}$$

$$m_{ij} z_{ij} \le x_{ij} \le M_i z_{ij},\ i = 1, ..., m,\ j = 1, ..., n, \tag{5.15}$$

где m – число поставщиков; n – число потребителей; переменные z_{ij} являются индикаторами наличия поставки от поставшика i потребителю j, а переменные x_{ij} обозначают размер соответствующей поставки. A_j – минимальное количество продукта, требуемое потребителю j; m_{ij} – минимальное количество продукта, которое поставщик i готов доставить потребителю j; M_i – максимальное общее количество продукта, которое поставщик i может доставить. В целевую функцию входят фиксированные доплаты a_{ij}, связанные с открытием поставки (i, j) и стоимости транспортировки единицы продукции c_{ij}. Параметры $a_{ij}, c_{ij}, m_{ij}, M_i, A_j$ предполагаются целочисленными и неотрицательными при всех i, j.

Введем обозначения для матриц переменных $\mathbf{Z} = (z_{ij}), \mathbf{X} = (x_{ij})$. Стоимость решения $(\mathbf{Z}, \mathbf{X})$, заданную выражением (5.11), будем обозначать через $f(\mathbf{Z}, \mathbf{X})$. Заметим, что при фиксированной матрице $\mathbf{Z}$ задача (5.11)–(5.15) становится задачей линейного программирования и может быть решена за полиномиально ограниченное время. Таким образом, при использовании $\mathbf{Z}$ для кодировки решений задача управления поставками соответствует определению задачи комбинаторной оптимизации (см. раздел 1.1).

С помощью сведения задачи о разбиении предметов легко показать, что задача нахождения допустимого решения, удовлетворяющего ограничениям (5.12)–(5.15), является NP-трудной даже в случае $n = 2$. Действительно, рассмотрим задачу о разбиении предметов: можно ли разбить множество натуральных чисел $\{p_1, p_2, \ldots, p_{m'}\}$ на два подмножества с равными суммами? Если сумма $\sum_{i=1}^{m'} p_i$ нечетна, то ответ, очевидно, отрицательный.

В противном случае положим $n = 2,\ m = m',\ m_{i1} = m_{i2} = M_i = p_i,\ i = 1, \ldots, m'$, и $A_1 = A_2 = \frac{1}{2}\sum_{i=1}^{m'} p_i$. При этом система (5.12)–(5.15) разрешима тогда и только тогда, когда требуемое разбиение существует.

Многие практически значимые задачи логистики, составления расписаний и планирования производства [190, 202, 218, 284, 285] могут быть сформулированы как частный случай задачи управления поставками.

Если $m_{ij} = 0$ при всех i, j и $\sum_{i=1}^{m} M_i = \sum_{j=1}^{n} A_j$, то данная задача представляет собой известную транспортную задачу с фиксированными доплатами – см., например, [9, 22, 123, 285]. Точному и приближенному решению последней задачи посвящен ряд публикаций. Точные алгоритмы, как правило, базируются на методах ветвей и границ с отсечениями [9,123,196,255]. Среди эвристических методов, разработанных для транспортной задачи с фиксированными доплатами, известны алгоритмы поиска с запретами [151, 285], генетические алгоритмы [170, 284] и другие подходы. В большинстве эвристических алгоритмов используется тот факт, что множество вершин многогранника линейной релаксации задачи содержит оптимальное решение. Однако в общем случае для задачи управления поставками это свойство не выполняется (достаточно рассмотреть пример задачи, где $m = n = 1$, $a_{11} = c_{11} = m_{11} = M_1 = 1$, $A_1 = 1/2$).

Задача (5.11)–(5.15) является ослабленной версией задачи управления поставками, рассмотренной в [142,145]: в этих работах в условии (5.12) требуется равенство, а при открытой поставке ее стоимость является вогнутой неубывающей функцией от размера поставки. Нахождение любого допустимого решения задачи о поставках продукции в формулировке из [145] является NP-трудной задачей даже при одном потребителе, хотя при целочисленных исходных данных существует псевдополиномиальный точный алгоритм ее решения [142]. Некоторые важные свойства задачи управления поставками в постановке из [142, 145] могут быть перенесены и на задачу (5.11)–(5.15). Например, можно показать, что в любой разре-

шимой индивидуальной задаче с одним потребителем имеется оптимальное решение, где $x_{i1} \in \{0, m_{i1}, M_{i1}\}$ для всех поставщиков i, кроме, быть может, одного из них [44], а генетический алгоритм с жадным декодером (п 5.2.3) имеет общую схему, подобную схеме ГА, разработанного П.А. Борисовским в [13] для задачи управления поставками с равенством в условии (5.12).

Как будет показано в п. 5.2.2, задача (5.11)–(5.15) представляет больше перспектив для поиска приближенных решений в случае одного потребителя, чем задача в постановке из [142,145]. Запишем задачу (5.11)–(5.15) при $n = 1$ в компактном виде:

$$\min \quad \sum_{i=1}^{m} k_i(x_i), \tag{5.16}$$

$$\sum_{i=1}^{m} x_i \geq A, \tag{5.17}$$

$$x_i \in \{0\} \cup [m_i, M_i], \quad i = 1, \ldots, m. \tag{5.18}$$

Здесь переменными являются только объемы поставок x_i, $i = 1, \ldots, m$, а стоимости доставки продукции от поставщика i выражены функциями

$$k_i(x_i) = \begin{cases} 0, & \text{если } x_i = 0; \\ a_i + c_i x_i, & \text{если } x_i > 0, \end{cases} \tag{5.19}$$

где a_i, c_i, m_i, M_i, A – целые неотрицательные числа при $i = 1, \ldots, m$. Подобно общему случаю будем обозначать стоимость решения $\mathbf{x} = (x_1, \ldots, x_m)$, заданную выражением (5.16), через $f(\mathbf{x})$.

Последняя задача является NP-трудной, так как она содержит как частный случай задачу об одномерном булевом рюкзаке (см. раздел 1.1). Для того чтобы в этом убедиться, достаточно положить m равным числу переменных в задаче о рюкзаке, $m_i = M_i = a_i$, $i = 1, \ldots, m$, и оставить те же значения коэффициентов c_i, $i = 1, \ldots, m$, что и в задаче о рюкзаке.

В п. 5.2.2 описывается жадный алгоритм [44] трудоемкости $O(m^2)$ для задачи управления поставками при $n = 1$, здесь же доказано, что данный алгоритм является 2-приближенным, если стоимость поставки $k_i(x_i)$

от каждого поставщика i является вогнутой неубывающей функцией на отрезке $[0, M_i]$. В [41] нами предложена вполне полиномиальная аппроксимационная схема для случая $n = 1$ в более общей постановке, где предполагается, что объем поставки от каждого поставщика принадлежит объединению конечного числа отрезков, функции стоимости поставок являются вогнутыми и неубывающими в пределах каждого из интервалов.

В [144] построена вполне полиномиальная аппроксимационная схема, пригодная для более широкого класса целевых функций и основанная на подходе, изложенном в [293]. В применении к случаю, когда функции стоимости поставки имеют вид (5.19), эта схема имеет временную сложность

$$O((m/\varepsilon)^2 \log_2(mk_{\max}) \log_2(k_{\max})), \tag{5.20}$$

где $k_{\max} = \max\limits_{i=1,\ldots,m} k_i(M_i)$. Несколько позднее близкая по свойствам вполне полиномиальная аппроксимационная схема была предложена С.Т. Нг, М.Я. Ковалевым и Т.С.Е. Ченгом в [250]. Во многих случаях трудоемкость алгоритма [250] оказывается ниже трудоемкости алгоритма из [144].

Докажем упомянутое выше структурное свойство для случая $n = 1$.

Утверждение 5.1. *Разрешимая задача* (5.16)–(5.19) *имеет оптимальное решение* $\mathbf{x} \in \mathbb{Z}^m$, *где* $x_i \in \{0, m_i, M_i\}$ *при всех* $i = 1, \ldots, m$, *кроме, может быть, одного.*

Доказательство. Рассмотрим некоторое оптимальное решение $\mathbf{x}^* \in \mathbb{Z}^m$ задачи (5.16)–(5.19). Решение $\mathbf{x}^*$ существует ввиду того, что при фиксированных переменных (z_{ij}) задача (5.16)–(5.19) является задачей линейного программирования с целочисленным многогранником.

Пусть найдется такая пара индексов i, j, $1 \le i, j \le m$, что $m_i < x_i^* < M_i$, $m_j < x_j^* < M_j$ и $c_i \le c_j$. Положим $\delta = \min\{M_i - x_i^*, x_j^* - m_j\}$, $y_i = x_i^* + \delta$ и $y_j = x_j^* - \delta$; остальные координаты решений $\mathbf{y}$ и $\mathbf{x}^*$ совпадают. Легко видеть, что $\mathbf{y}$ также является оптимальным решением и $y_i = M_i$ либо $y_j = m_j$. Из целочисленности δ следует целочисленность $\mathbf{y}$ и $\mathbf{x}^*$. $\square$

5.2.1. Сложность приближенного решения в случае нескольких потребителей

Покажем, что поиск приближенного решения с какой-либо константной оценкой точности для общего случая задачи (5.11)–(5.15) представляет собой NP-трудную в сильном смысле задачу. Для этого будем использовать задачу распознавания наличия вершинного покрытия требуемой мощности K в графе $G = (V, E)$ с множеством вершин $V = \{1, 2, \ldots, q\}$ и множеством ребер $E = \{e_1, e_2, \ldots, e_r\}$. Пусть $\deg j$ – степень вершины j.

Опишем сведение ЗВП к задаче (5.11)–(5.15). Поставим в соответствие вершинам графа G пункты потребления, а ребрам – пункты производства и введем дополнительный пункт производства с номером $r + 1$. Таким образом, имеем $m = r + 1$ и $n = q$. В данном сведении открытие поставок от пункта производства с номером $r + 1$ будет соответствовать включению вершин в покрытие. Спрос A_j каждого потребителя j положим равным $\deg j$.

Для каждого поставщика $i,\ i = 1, \ldots, r$, назначим $M_i = m_{ij} = 1$, $c_{ij} = 0$ при всех j, и пусть $a_{ij} = 0$ для двух его «выделенных» потребителей, соответствующих вершинам инцидентным ребру e_i в графе G. Для остальных потребителей $j \notin e_i$ полагаем $a_{ij} = \Omega$, где $\Omega > q$ – достаточно большое число. Другими словами, каждый поставщик $i,\ i = 1, \ldots, r$, может отправить единицу продукции по нулевой цене только одному из своих двух «выделенных» потребителей. Для дополнительного поставщика $r + 1$ положим $m_{r+1,j} = q,\ a_{r+1,j} = 1,\ c_{r+1,j} = 0$ при всех j; $M_{r+1} = Kq$.

Далее полученную с помощью описанной сводимости задачу о поставках будем обозначать через $P(G)$.

Лемма 5.1. a) *Если G имеет вершинное покрытие мощности не более K, то оптимальное значение целевой функции в $P(G)$ не превышает K.*
b) *Если задача $P(G)$ имеет допустимое решение стоимости менее Ω, то*

существует вершинное покрытие графа G мощности не более K.

Доказательство. (a) Пусть C – вершинное покрытие графа G. Положим $z_{r+1,j} = 1$ для каждого j и размер поставки от дополнительного поставщика равным q, если $j \in C$; в противном случае $x_{r+1,j} = z_{r+1,j} = 0$. Тогда, для того чтобы $(\mathbf{Z}, \mathbf{X})$ было допустимым решением задачи (5.11)–(5.15), достаточно для всех $i \leq r$ и всех j положить $x_{ij} = 1,\ z_{ij} = 1$ при $j \in e_i$, $j \notin C$; в противном случае $x_{ij} = 0,\ z_{ij} = 1$. Так как все поставки от поставщиков $i = 1, 2, \ldots, r$ направлены только «выделенным» потребителям и поставщик $r+1$ обслуживает не более K клиентов, то $f(\mathbf{Z}, \mathbf{X}) \leq K$.

(b) Пусть $(\mathbf{Z}, \mathbf{X})$ – допустимое решение задачи (5.11)–(5.15) и $f(\mathbf{Z}, \mathbf{X}) < \Omega$. Обозначим $C(\mathbf{X}) = \{j | x_{r+1,j} > 0\}$. Здесь $|C(\mathbf{X})| \leq K$, так как $x_{r+1,j} \geq q$ для всех $j \in C(\mathbf{X})$, и $x_{r+1,1} + \ldots + x_{r+1,m} \leq M_{r+1} = Kq$. Первые r поставщиков снабжают только «выделенных» потребителей, поэтому $C(\mathbf{X})$ – вершинное покрытие, так как если $x_{r+1,u} = 0$ и $x_{r+1,v} = 0$ для какого-либо ребра $e_i = uv$, то $\sum\limits_{i=1}^{m} x_{iu} + \sum\limits_{i=1}^{m} x_{iv} \leq \deg u + \deg v - 1 < A_u + A_v$, что противоречит (5.12). □

Теорема 5.1. *Нахождение $p(m, n)$-приближенного решения задачи управления поставками продукции при любом полиноме $p(m, n)$ представляет собой NP-трудную в сильном смысле задачу.*

Доказательство. Пусть дан граф G и $\Omega = qp(r, q)$. В соответствующей индивидуальной задаче о поставках $P(G)$ все численные параметры ограничены некоторым полиномом от длины входа ЗВП графа G.

Пусть $(\mathbf{Z}^*, \mathbf{X}^*)$ – оптимальное решение задачи (5.11)–(5.15), а $(\mathbf{Z}', \mathbf{X}')$ – ее $p(r, q)$-приближенное решение. С одной стороны, если существует вершинное покрытие мощности не более K, то по утверждению (a) леммы 5.1 имеем $f(\mathbf{Z}^*, \mathbf{X}^*) \leq K$, следовательно, $f(\mathbf{Z}', \mathbf{X}') \leq Kp(r, q) < \Omega$. С другой стороны, если $f(\mathbf{Z}', \mathbf{X}') \leq Kp(r, q)$, то $f(\mathbf{Z}^*, \mathbf{X}^*) < \Omega$ и покрытие мощности не более K существует по утверждению (b) леммы 5.1.

Таким образом, наличие $p(m,n)$-приближенного решения $(\mathbf{Z}', \mathbf{X}')$ для задачи (5.11)–(5.15) позволяет ответить на вопрос о существовании требуемого вершинного покрытия положительно, если $f(\mathbf{Z}', \mathbf{X}') < \Omega$, и отрицательно – в противном случае. □

5.2.2. 2-приближенный жадный алгоритм для случая одного потребителя

В данном подпункте предлагается жадный алгоритм для задачи (5.16)–(5.18), когда стоимости $k_i(x_i),\ i = 1, \ldots, m$, заданы функциями

$$k_i(x_i) = \begin{cases} 0, & \text{если } x_i = 0; \\ a_i + g_i(x_i), & \text{если } x_i > 0, \end{cases} \tag{5.21}$$

где $a_i \geq 0$ при любом i, а функции $g_i(x_i)$ являются неотрицательными и неубывающими при $x_i \in [m_i, M_i]$. Относительно входных данных $a_i, m_i, M_i,\ i = 1, \ldots, m$, здесь будем предполагать только то, что они являются неотрицательными рациональными числами. Обозначим через $\mathbf{y} = (y_1, \ldots, y_m)$ приближенное решение задачи (5.16)–(5.18), (5.21), найденное жадным алгоритмом; ℓ – текущее количество продукта, которое еще необходимо доставить. На каждой итерации жадного алгоритма добавляется наиболее дешевая поставка в относительном выражении.

Алгоритм 5.4. Жадный алгоритм

1. Положить $y_k := 0,\ k = 1, 2, \ldots, m$.

2. Положить $\ell := A - \sum\limits_{k=1}^{m} y_k$ и выбрать такое j, что $y_j = 0$ и

$$\frac{k_j(\min\{\max\{\ell, m_j\}, M_j\})}{\min\{\ell, M_j\}} \leq \frac{k_s(\min\{\max\{\ell, m_s\}, M_s\})}{\min\{\ell, M_s\}}, \tag{5.22}$$

для всех s таких, что $y_s = 0$.

Если $y_j > 0$ для всех j, то конец работы.

3. Положить $y_j := \min\{\max\{\ell, m_j\}, M_j\}$.

4. Если $\ell - y_j > 0$, перейти на шаг 2.

Конец работы.

Если после остановки алгоритма выполняется неравенство $\sum\limits_{k=1}^{m} y_k < A$, то задача не имеет допустимых решений. Пусть T_g – верхняя граница трудоемкости вычисления функции $g_i(x_i)$ при всех i и рациональных $x_i \in [m_i, M_i]$. Тогда, как легко видеть, трудоемкость алгоритма есть $O(m^2 T_g)$.

Теорема 5.2. *Жадный алгоритм является 2-приближенным алгоритмом для задачи* (5.16)–(5.18) при вогнутых функциях $k_i(x_i)$ вида (5.21).

Доказательство. Заметим, что на шаге 3 переменная y_j принимает значение M_j на всех итерациях, кроме последней; на ней y_j равна ℓ или m_j (здесь и далее под ℓ понимается значение, которое величина ℓ принимает по окончании выполнения алгоритма). Не ограничивая общности, после завершения работы алгоритма перенумеруем координаты решения так, что при некотором i выполнено равенство $y_j = M_j$ при всех $j \le i$; $y_{i+1} < M_{i+1}$ и $y_j = 0$ при всех $j > i+1$. Пусть f^* – стоимость оптимального решения задачи (5.16)–(5.18), (5.21). Положим

$$h(\mathbf{x}) = \sum_{j=1}^{m} \frac{(a_j + g_j(M_j))x_j}{M_j}$$

и рассмотрим задачу минимизации функции $h(\mathbf{x})$ на множестве, задаваемом ограничениями (5.17)–(5.18). Рассматривая оптимум $\hat{\mathbf{x}}$ линейной релаксации данной задачи, т. е. задачи, в которой условие (5.18) заменено на $0 \le x_j \le M_j,\ j = 1, \ldots, m$, заключаем, что

$$h(\hat{\mathbf{x}}) \ge \sum_{j=1}^{i} a_j + g_j(M_j) = \sum_{j=1}^{i} k_j(M_j).$$

По определению $h(\mathbf{x})$ имеем $h(\mathbf{x}) \le f(\mathbf{x})$ для любого $\mathbf{x}$, удовлетворяющего условиям (5.17)–(5.18). Таким образом,

$$f^* \ge \sum_{j=1}^{i} k_j(M_j).$$

Следовательно, если $y_{i+1} = 0$ (т. е. $\ell = M_i$ на последней итерации жадного алгоритма), то $\mathbf{y}$ – оптимальное решение.

Теперь убедимся в том, что если $y_{i+1} > 0$, то $f^* \geq k_{i+1}(y_{i+1})$. Действительно, пусть существует допустимое решение $\mathbf{y}'$ такое, что

$$\sum_{j=1}^{m} k_j(y'_j) < k_{i+1}(y_{i+1}). \tag{5.23}$$

Так как $\ell = A - \sum\limits_{k=1}^{i} M_k$, то

$$\sum_{r>i} y'_r \geq A - \sum_{j\leq i} y'_j \geq \ell. \tag{5.24}$$

Умножая (5.23) на $\ell/k_{i+1}(y_{i+1})$ и убирая первые i слагаемых, получаем

$$\sum_{r>i} \frac{\ell k_s(y'_r)}{k_{i+1}(y_{i+1})} < \ell. \tag{5.25}$$

Жадное правило (5.22), примененное на последней итерации, при всех $r > i$ обеспечивает неравенство

$$\frac{k_{i+1}(y_{i+1})}{\ell} = \frac{k_{i+1}(y_{i+1})}{\min\{\ell, M_{i+1}\}} \leq \frac{k_r(\min\{\max\{\ell, m_r\}, M_r\})}{\min\{\ell, M_r\}}. \tag{5.26}$$

Таким образом, из (5.25) следует неравенство

$$\sum_{r>i} \frac{\min\{\ell, M_r\} k_r(y'_r)}{k_r(\min\{\max\{\ell, m_r\}, M_r\})} < \ell. \tag{5.27}$$

Рассмотрим слагаемые этой суммы с $y'_r > 0$. Если $\ell < m_r$, то из (5.26) имеем $k_r(m_r)/\ell \geq k_{i+1}(y_{i+1})/\ell$, что противоречит неравенству $k_r(y'_r) < k_{i+1}(y_{i+1})$, полученному из (5.23), поэтому слагаемые с $\ell < m_r$ отсутствуют в (5.27).

Если $m_r \leq \ell \leq M_r$, то соответствующие слагаемые в (5.27) принимают вид $\ell k_r(y'_r)/k_r(\ell)$. Возможны два случая:

(a) $y'_r \geq \ell$. Так как $k_r(x)$ не убывает, то $k_r(y'_r) \geq k_r(\ell)$. Из (5.26) имеем $k_r(\ell) \geq k_{i+1}(y_{i+1})$. Следовательно, $k_r(y'_r) \geq k_{i+1}(y_{i+1})$, что противоречит (5.23), поэтому слагаемые с $y'_r \geq \ell$ также отсутствуют в (5.27).

(b) $y'_r < \ell$. Из вогнутости функции k_r следует, что $\ell k_r(y'_r)/k_r(\ell) \geq y'_r$.

Наконец, если $\ell > M_r$, слагаемые в (5.27) превращаются в $M_r k_r(y'_r)/k_r(M_r)$, и в силу вогнутости функции k_r, $M_r k_r(y'_r)/k_r(M_r) \geq y'_r$. Следовательно,

$$\sum_{r>i} y'_r \leq \sum_{r>i} \frac{\min\{\ell, M_r\} k_r(y'_r)}{k_r(\min\{\max\{\ell, m_r\}, M_r\})}, \tag{5.28}$$

и из (5.27) и (5.28) получаем неравенство $\sum\limits_{r>i} y'_r < \ell$, что противоречит (5.24). Таким образом, $f^* \geq k_{i+1}(y_{i+1})$ и

$$2f^* \geq \sum_{j=1}^{i} k_j(M_j) + k_{i+1}(y_{i+1}).$$

□

5.2.3. Генетический алгоритм с жадным декодером

Построим генетический алгоритм GAgrd на основе недвоичной кодировки решений, где пробное решение вычисляется по заданному генотипу посредством решения серии вспомогательных задач управления поставками с одним потребителем. В алгоритме декодирования потребители рассматриваются поочередно и каждому из них назначаются поставки из оставшегося у поставщиков объема продукции. При этом генотип представляет собой перестановку $\xi = (j_1, ..., j_n)$ элементов $1, \ldots, n$.

При декодировании фрагмента решения, относящегося к потребителю с номером $j = j_k \in \{1, ..., n\}$, рассматривается следующий релаксированный вариант задачи. Воспользуемся компактными обозначениями для входных данных $A = A_j, c_i = c_{ij}, a_i = a_{ij}, m_i = m_{ij}$ и переменных $x'_i = x_{ij}, z'_i = z_{ij}$. Пусть $M'_i \in \mathbb{Z}$, $i = 1, \ldots, m$, обозначает объем продукции, имеющийся у поставщика i за вычетом объема его поставок, которые уже были назначены при декодировании фрагментов решения, относящихся к потребителям $j_1, \ldots, j_{k-1}$. Релаксированная задача управления поставками

с одним потребителем формулируется следующим образом:

$$\min \sum_{i=1}^{m}(a_i z'_i + c_i x'_i) + R\alpha, \tag{5.29}$$

$$\sum_{i=1}^{m} x'_i \geq A - \alpha, \tag{5.30}$$

$$m_i z_i \leq x'_i \leq M'_i z'_i,\ i = 1, ..., m, \tag{5.31}$$

$$\alpha \in \mathbb{R}_+,\ z'_i \in \{0, 1\},\ x'_i \in \mathbb{R}_+,\ i = 1, ..., m. \tag{5.32}$$

Параметр R представляет собой штрафной коэффициент, а переменная невязки α допускает некоторое нарушение ограничений исходной задачи. Заметим, что в разрешимой задаче управления поставками при целочисленных входных данных существует целочисленное допустимое решение (см. утверждение 5.1). Следовательно, параметр R может быть выбран достаточно большим, так что значение критерия (5.29) для любого допустимого решения задачи (5.29)–(5.32) с $\alpha = 0$ было меньше значения этого критерия в любом решении с ненулевой невязкой.

Пробное решение $(\mathbf{Z}, \mathbf{X}) = \Psi(\xi)$ вычисляется по генотипу $\xi = (j_1, ..., j_n)$ с использованием следующего алгоритма:

Алгоритм 5.5. Декодирование перестановки

1. Положить $M'_i := M_i,\ i = 1, ..., m$.

2. Для $k := 1$ до n выполнять:

2.1. Решить задачу (5.29) – (5.32) для одного потребителя j_k.

Пусть получено решение $((x'_1, \ldots, x'_m),\ (z'_1, \ldots, z'_m))$.

Определить фрагмент решения задачи (5.11) – (5.15):

$$x_{ij_k} = x'_i,\ z_{ij_k} = z'_i,\ i = 1, \ldots, m.$$

2.2. Пересчитать остаточные объемы продукции: $M'_i := M'_i - x'_i$.

3. Применить к $(\mathbf{Z}, \mathbf{X})$ рандомизированный алгоритм локального поиска.

Для решения задачи с одним потребителем на шаге 2.1 сначала применяется жадный алгоритм 5.4. Если по его завершении окажется что в полученном решении $(x'_1, ..., x'_m)$ общий объем поставок выбранному потребителю превышает требуемый, т. е. $\sum_{i=1}^{m} x'_i > A$, то применяется процедура удаления излишнего объема поставок. В этой процедуре сначала по возможности исключаются поставки малого объема, затем максимально сокращается объем открытых поставок:

Алгоритм 5.6. Процедура удаления излишнего объема поставок

1. Положить $A' := A - \sum_{i=1}^{m} x'_i$.

2. Пока $A' < 0$ и $m_i \leq x'_i \leq -A'$ для некоторого i, выполнять:

2.1 Выбрать i, такое что $m_i \leq x'_i \leq -A'$, максимизируя $a_i + c_i x'_i$.

2.2 Положить $A' := A' + x'_i$ и $x'_i := 0$.

3. Пока $A' < 0$ и $x'_i > m_i$ для некоторого i, выполнять:

3.1 Выбрать i, такое что $x'_i > m_i$, максимизируя c_i.

3.2 Положить $d := \min\{x'_i - m_i, -A'\}$, $x'_i := x'_i - d$, и $A' := A' + d$.

По завершении жадного алгоритма может оказаться, что полученное решение является недопустимым с точки зрения исходной задачи, т. е. $\sum_{i=1}^{m} x'_i < A$. С учетом выбора параметра R это означает, что текущая вспомогательная задача не имеет допустимых решений без штрафа.

Вместо жадного алгоритма в процедуре декодирования может быть применен любой другой приближенный алгоритм для задачи с одним потребителем. При использовании аппроксимационной схемы (см., например, [44, 143, 144]) имеет смысл снижать параметр ϵ в процессе работы ГА.

Заметим, что не каждое решение задачи (5.11)–(5.15) может быть получено в результате декодирования подходящей перестановки ξ. Более того, можно привести пример разрешимой индивидуальной задачи, для которой даже при вычислении оптимальных решений для каждой подзада-

чи вида (5.29)–(5.32), результат работы декодера не является допустимым решением ни при каком генотипе ξ. Действительно, рассмотрим задачу управления поставками размерности 2×2, где

$$A_1 = A_2 = 3,\ M_1 = 4,\ M_2 = 2 \ \text{ и } \ (c_{ij}) = \begin{pmatrix} 1 & 1 \\ 2 & 2 \end{pmatrix}, (m_{ij}) = \begin{pmatrix} 2 & 2 \\ 1 & 1 \end{pmatrix}.$$

Единственное допустимое решение данной задачи имеет компоненты $x_{ij} = m_{ij},\ i = 1, 2,\ j = 1, 2$. Поэтому если для первого потребителя подзадачу (5.29)–(5.32) решить оптимально, положив $x_{11} = 3, x_{21} = 0$, то требуемый объем для второго потребителя не может быть доставлен. Аналогичные рассуждения применимы и в случае, если сначала оптимальным образом решается подзадача второго потребителя.

На шаге 3 в алгоритме декодирования 5.5 используется следующая процедура рандомизированного локального поиска. На каждой итерации данной процедуры случайным образом выбираются четыре элемента матрицы (x_{ij}) так, чтобы по крайней мере два из них были ненулевыми. Далее для четырех выбранных элементов объемы поставок выбираются оптимальным образом (задача 2×2 легко решается аналитически). Если в результате удается сократить значение штрафа или улучшить значение целевой функции, то выбранным элементам матрицы (x_{ij}) присваиваются полученные значения. Процедура локального поиска состоит в выполнении заданного числа итераций описанного вида. В вычислительных экспериментах, представленных в п. 5.2.5 число итераций данной процедуры полагалось равным 100. Функция приспособленности вычисляется по формуле

$$\Phi(\xi) = \left(\sum_{j=1}^{n} \sum_{i=1}^{m} (a_i z_{ij} + c_i x_{ij}) + R \sum_{j=1}^{n} \min \left\{ 0, A_j - \sum_{i=1}^{m} x_{ij} \right\} \right)^{-1},$$

где пара $(\mathbf{X}, \mathbf{Z})$ получена в результате декодирования генотипа ξ.

Особенности генетического алгоритма GAgrd. Генетический алгоритм GAgrd реализован с использованием стационарной стратегии управ-

ления популяцией и турнирной селекции с размером турнира s. Начальная популяция состоит из N генотипов, представляющих собой равновероятно выбранные перестановки.

Выбор особей для удаления (шаг 6 алгоритма 1.2) осуществляется с помощью турнирной селекции с размером турнира s. При этом вместо значений приспособленности особей используются обратные к ним величины (чем меньше приспособленность особи, тем больше вероятность того, что особь будет удалена из популяции). Критерием остановки данного ГА является достижение предела на общее время вычислений. Алгоритм независимо выполняется K раз, и на каждое его выполнение выделяется время, равное T/K, где T – общее время, отведенное для вычислений.

Операторы мутации и кроссинговера. Для выполнения мутации используется *оператор обмена*, где в заданном генотипе $\xi = (\xi_1, ..., \xi_n)$ выбирается пара генов со случайными номерами k и k' и их значения ξ_k и $\xi_{k'}$ меняются местами. Оператор обмена выполняется с заданной вероятностью p_{x}, иначе генотип ξ проходит процедуру мутации без изменений.

В GAgrd используется *оператор кроссинговера с частичным отображением* [200], обозначаемый через PMX (от английского *partially mapped crossover*). Данный оператор применяется с заданной вероятностью p_{cross}.

Как показали эксперименты, кроссинговер PMX показывает хорошие результаты в ряде задач составления расписаний, в то время как для задачи коммивояжера лучшие результаты показали операторы, основанные на наследовании свойства смежности вершин [269].

5.2.4. Генетический алгоритм с ЦЛП-кроссинговером

Как отмечалось при обсуждении постановки задачи (5.11)–(5.15), для кодировки решений задачи управления поставками достаточно указать лишь только двоичную матрицу $\mathbf{Z}$. В предлагаемом ГА с ЦЛП-

кроссинговером генотип представляет собой матрицу $\mathbf{Z}$.

Объемы поставок x_{ij} при заданной матрице $\mathbf{Z}$ могут вычисляться посредством решения транспортной задачи с нижними и верхними ограничениями на объемы перевозок. Однако последняя задача может не иметь допустимых решений. Для того чтобы учесть такую возможность, рассмотрим следующий релаксированный вариант исходной задачи, где часть ограничений представлена в виде штрафных слагаемых в целевой функции:

$$\min\ C_0 + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + R\left(\sum_{j=1}^{n}\alpha_j + \sum_{i=1}^{m}\beta_i\right), \tag{5.33}$$

$$\sum_{i=1}^{m} x_{ij} \geq A_j - \alpha_j,\ j = 1, ..., n, \tag{5.34}$$

$$\sum_{j=1}^{n} x_{ij} \leq M_i + \beta_i,\ i = 1, ..., m, \tag{5.35}$$

$$m_{ij}z_{ij} \leq x_{ij} \leq M_i z_{ij},\quad i = 1, ..., m,\ j = 1, ..., n. \tag{5.36}$$

$$\alpha_j \geq 0,\ j = 1, ..., n,\quad \beta_i \geq 0,\ i = 1, ..., m. \tag{5.37}$$

Здесь $C_0 = \sum_{i=1}^{m}\sum_{j=1}^{n} z_{ij}a_{ij}$ – величина фиксированных затрат, связанных с генотипом $\mathbf{Z}$. При достаточно большом значении штрафного коэффициента R слагаемое $\sum_{j=1}^{n}\alpha_j + \sum_{i=1}^{m}\beta_i$ в оптимальном решении задачи (5.33)–(5.37) обращается в 0 всякий раз, когда генотипу $\mathbf{Z}$ соответствует некоторый набор переменных (x_{ij}), являющийся допустимым в задаче (5.11)–(5.15). Пусть функция приспособленности имеет вид $\Phi(\mathbf{Z}) = 1/\tilde{f}(\mathbf{Z})$, где $\tilde{f}(\mathbf{Z})$ – значение целевой функции (5.33) в оптимальном решении задачи (5.33)–(5.37) при заданном генотипе $\mathbf{Z}$.

Оператор ЦЛП-кроссинговера

При разработке оператора рекомбинации преследовались две цели. Первая цель состояла в построении оператора кроссинговера, в котором

задача оптимальной рекомбинации решается с достаточно низкой трудоемкостью. Второй целью являлось внесение случайных изменений в родительские генотипы, подобно тому, как это происходит в известных операторах мутации. Исходя из этих целей сформулируем вспомогательную задачу ЧЦЛП на основе ЗОР или ослабленной ЗОР (см. определение 3.1 и условие (3.2)) для задачи (5.11)–(5.15).

Вспомогательная задача ЦЛП-кроссинговера на основе ЗОР представляет собой задачу (5.11)–(5.15) с дополнительным условием «фиксации» переменных $z_{ij} = p^1_{jq}$ при всех i, j, таких что $p^1_{ij} = p^2_{ij}$, кроме некоторого случайного подмножества переменных S_{mut}:

$$z_{ij} = p^1_{ij} \;\text{ для всех } i,j:\; p^1_{ij} = p^2_{ij} \text{ и } (i,j) \notin S_{\text{mut}}. \tag{5.38}$$

Аналогичным образом, если за основу берется ослабленная ЗОР, то «фиксируются» переменные z_{ij}, имеющие *нулевое* значение в обоих родительских генотипах $\mathbf{p}^1, \mathbf{p}^2$, кроме случайного подмножества переменных S_{mut}:

$$z_{ij} = 0 \;\text{ для всех } i,j:\; p^1_{ij} = p^2_{ij} = 0 \text{ и } (i,j) \notin S_{\text{mut}}. \tag{5.39}$$

Подмножество S_{mut} формируется следующим образом. Всякая пара индексов $(i,j) : p^1_{ij} = p^2_{ij}$ независимо от других таких пар включается во множество S_{mut} с заданной вероятностью P_{m}.

Если найдено оптимальное или приближенное решение вспомогательной задачи, то это решение $\mathbf{Z}_{\text{mipr}}$ становится результатом кроссинговера.

Как показывает утверждение 5.2, построение ЦЛП-кроссинговера полиномиальной трудоемкости не представляется возможным.

Утверждение 5.2. *ЗОР и ослабленная ЗОР для задачи управления поставками продукции являются NP-трудными даже в случае одного потребителя.*

Доказательство. Рассмотрим сводимость задачи об одномерном булевом рюкзаке к задаче управления поставками продукции. Пусть вход

задачи об одномерном булевом рюкзаке составляют неотрицательные целые числа $a'_i, c'_i,\ i = 1, \ldots, n'$ и A'. Требуется найти набор предметов $K \subseteq \{1, \ldots, n'\}$, максимизирующий сумму $\sum_{i \in K'} c'_i$, при условии $\sum_{i \in K} a'_i \leq A'$.

Положим $n = 1,\ m = n',\ m_i = M_i = a'_i,\ i = 1, \ldots, n'$, и сохраняются те же значения коэффициентов $c_i,\ i = 1, \ldots, n'$, что и в задаче о рюкзаке. Данная сводимость показывает, что с точностью до обозначений рассматриваемый класс индивидуальных задач управления поставками продукции совпадает с множеством индивидуальных задач об одномерном булевом рюкзаке и способы кодировки решений одинаковы. В таком случае ЗОР (ослабленная ЗОР) для задачи управления поставками продукции не может быть проще, чем ЗОР (ослабленная ЗОР) для задачи о рюкзаке. Справедливость доказываемого утверждения вытекает из NP-трудности ЗОР (ослабленной ЗОР) для задачи о рюкзаке (см. теорему 3.3). □

Отыскание матрицы $\mathbf{Z}_{\text{mipr}}$ может быть реализовано, например, с помощью метода ветвей и границ с отсечениями. В частности, в п. 5.2.5 предлагается решать ослабленную задачу оптимальной рекомбинации (5.11)–(5.15), (5.39) с помощью решателя задач ЧЦЛП из пакета CPLEX 9.0. Во избежание чрезмерных вычислительных затрат при решении данной задачи устаналивается предельная длительность вычислений T_{rec} при каждом обращении к решателю из пакета CPLEX.

В случае если за отведенное время не удается найти оптимальное или хотя бы допустимое решение подзадачи, то оператор кроссинговера возвращает генотип $\text{Mut}^*(\mathbf{y}^1)$, вычисленный с помощью оператора мутации классического ГА (см. п. 1.2.1). Те же действия выполняются, если найденный генотип потомка $\mathbf{Z}_{\text{mipr}}$ уже присутствует в текущей популяции.

Генетический алгоритм со стационарной схемой управления популяцией (см. алгоритм 1.2), ЦЛП-кроссинговером и функцией приспособленности $\Phi(\mathbf{Z}) = 1/\tilde{f}(\mathbf{Z})$ далее обозначается через GAmipr. При этом на шаге 5 алгоритма 1.2 оператор Mut действует как тождественное отображение. По

умолчанию в данном алгоритме предполагается ЦЛП-кроссинговер, основанный на ослабленной формулировке ЗОР (базовый вариант GAmipr). При упоминании GAmipr с ЦЛП-кроссинговером, основанным на ЗОР согласно определению 3.1, это отличие будет специально указываться как модифицированный вариант GAmipr. Генотипы начальной популяции в этом ГА генерируются тем же способом, что и в GAgrd (см. п. 5.2.3). Рассматривался также и другой способ выбора начальных генотипов, где каждый бит принимает значение 1 с заданной вероятностью. Однако все опробованные способы выбора такой вероятности оказались практически неприемлемыми из-за высокой вероятности получения недопустимого решения.

В тех случаях, когда при построении потомка применяется оператор мутации Mut^*, для вычисления приспособленности потомка $\Phi(\mathbf{Z})$ необходимо решить задачу линейного программирования (5.33) – (5.37). Эту же задачу требуется решать для вычисления приспособленности, когда при построении начальной популяции возникают недопустимые генотипы. В экспериментах в п. 5.2.5 для решения таких задач линейного программирования также используется пакет CPLEX 9.0.

В процессе работы GAmipr происходит адаптивная настройка вероятности мутации P_{m}. На первой итерации эта вероятность равна априорно заданной величине P_{m}^0. Всякий раз, когда удается найти оптимум в подзадаче (5.11) – (5.15), (5.39), параметр P_{m} умножается на 1.1. Если же оптимум в этой подзадаче за время T_{rec} не был найден и если при этом $P_{\mathrm{m}} > P_{\mathrm{m}}^0$, то параметр P_{m} снижается вдвое. Описанный способ настройки вероятности мутации позволяет получать подзадачи такой размерности, при которой алгоритм ветвей и границ находит их оптимум за время, близкое к T_{rec}.

Алгоритм GAmipr выполняется несколько раз с независимой инициализацией начальной популяции и параметра P_{m}. Обозначим через θ время работы ЦПУ, прошедшее с начала последней инициализации ГА до момента последнего улучшения рекордного значения приспособленности в популя-

Таблица 5.7. Размерности задач из серии MIS

задача	n	m	$\max a_{ij}$
hamming6.4	64	1313	4097
c125.9	125	788	15626
johnson16.2.4	120	1681	14401
mann_a27	378	703	142885
gen200_p0.9_44	200	1991	40001
gen200_p0.9_55	200	1991	40001

ции. Будем полагать, что условие остановки выполняется, если за последние θ единиц времени работы ЦПУ не было ни одного улучшения рекорда по приспособленности и, кроме того, $t > N$.

5.2.5. Экспериментальные результаты

Реализация алгоритмов и тестовые примеры. Алгоритм GAgrd запрограммирован в среде Visual C++ 6.0, а алгоритм GAmipr реализован с использованием языка OPL Script в среде ILOG OPL Studio 3.7. Кроме того, пакет решения задач частично целочисленного линейного программирования CPLEX 9.0 тестировался для поиска решения задачи как отдельный модуль. Эксперименты проводились со следующими сериями примеров.

- **Серия RAN** состоит из 8 примеров, построенных с помощью сведения транспортной задачи с фиксированными доплатами с известными размерностями m, n и $m_{ij} = 0$ при всех i, j. Исходные данные транспортных задач с фиксированными доплатами были построены в [203] с использованием модифицированного генератора задач NETGEN [123] и доступны по адресу *http://plato.la.asu.edu/ftp/fctp/*.

- **Серия H** состоит из 14 примеров большей размерности, построенных из транспортных задач с фиксированными доплатами h_0, h_1, ..., h_c и h_e (пример h_d пропущен, так как он оказался идентичен примеру h_c). Исходные задачи построены в [285] тем же генерато-

ром NETGEN при $m = 100, n = 50$, $m_{ij} = 0$, а значения фиксированных доплат a_{ij} выбраны из интервала от 6400 до 25600.

- **Серия S** состоит из 8 задач, построенных случайным образом при $m = n = 50$. Все $m_{ij} = 10$; значения $M_i \in [400, 600]$, $A_j \in [350, 550]$, $a_{ij} \in [500, 4500]$ и $c_{ij} \in [20, 1020]$ выбраны генератором псевдослучайных чисел с равномерным распределением. Таким образом, суммарный объем товара, доступного для поставки, в среднем на 10% превышает общий требуемый потребителями объем.

- **Серия P** состоит из 14 задач, построенных случайным образом при $n = m = 50$. Значения $M_i \in [12, 20]$, $A_j \in [10, 20]$, $a_{ij} \in [5, 10]$, $c_{ij} \in [10, 20]$, $m_{ij} \in [5, 10]$ выбраны генератором псевдослучайных чисел с равномерным распределением. Суммарный объем товара, доступного для поставки, в среднем несколько превышает общий требуемый потребителями объем. Длина интервалов $[m_{ij}, M_i]$ достаточно мала, что затрудняет поиск допустимого решения в этих задачах.

- **Серия MIS** состоит из 6 задач, полученных из тестовых примеров для задачи о независимом множестве из коллекции DIMACS [224] с использованием сводимости, предложенной в теореме 5.1. Исходные данные задач DIMACS доступны по адресу *ftp://dimacs.rutgers.edu/pub/challenge/graph/*. Задачи имеют размерности $m = |E| + 1$, $n = |V|$, где V и E – множество вершин и множество ребер в исходном графе. Значения m, n, а также диапазоны изменения коэффициентов a_{ij} приведены в Таблице 5.7.

Исходные данные описанных тестовых примеров доступны в библиотеке тестовых задач «Дискретные задачи размещения» по адресу *http://math.nsc.ru/AP/benchmarks/*.

Настройка алгоритмов и организация эксперимента. С целью сравнения алгоритмов в равных условиях для каждой задачи всем алгоритмам было отведено равное процессорное время T. Здесь и далее указывается время ЦПУ для ЭВМ Pentium-IV с тактовой частотой 3 ГГц 2.5 Гб ОЗУ. Для задач меньшей размерности из серий RAN и S полагаем $T = 2 \cdot 10^3$ с., а для прочих задач время счета составляет 10^4 с.

Описанные выше ограничения времени ЦПУ относятся и к пакету CPLEX при использовании его в качестве отдельного модуля. При этом в стандартные настройки параметров пакета были внесены следующие изменения: `RINSHeur = 100, diveType = 3, MIPEmphasis = 1.` Параметром `RINSHeur = 100` включается процедура локального поиска с окрестностью, индуцированной решением ЛП-релаксации [153], а параметр `diveType=3` направляет процесс ветвления таким образом, чтобы в первую очередь исследовать область пространства решений в окрестности текущего рекордного решения. Параметр `MIPEmphasis = 1` устанавливает больший приоритет на улучшение рекордных решений и меньший приоритет на доказательство оптимальности полученного решения. Далее такой набор значений параметров будем называть *эвристическими настройками* CPLEX.

В экспериментах с алгоритмом GAmipr выбран размер турнира $s = 10$, штрафной множитель $R = 10^{10}$ и начальная вероятность мутации $P_{\mathrm{m}}^0 = 10^{-3}$. Время, отведенное на каждое обращение к оператору ЦЛП-кроссинговера, составляет $T_{rec} = T/2000$, т. е., как правило, в процессе работы ГА будет выполнено несколько тысяч итераций. Параметры решателя CPLEX в данном алгоритме отличаются от эвристических настроек только одним параметром: `MIPEmphasis = 0` (улучшение рекордных решений и доказательство оптимальности имеют равный приоритет). Для всех задач выбрано $N = 500$, кроме тех примеров из серии MIS, где предел на объем используемой памяти в OPL Studio не позволял создать популяцию требуемого размера. В связи с этим выбрано $N = 100$ в за-

Таблица 5.8. Результаты для серии RAN

	f_{best}	f^h_{cplex}	f_{GAmipr}	f_{GAgrd}
ran13x13	3252*	**3252**	3271	3284
ran12x21	3664*	3675	**3664**	3671
ran14x18	3712*	3746	**3712**	3714
ran4x64	9711*	**9711**	**9711**	9726
ran8x32	5247*	**5247**	**5247**	**5247**
ran16x16	3823*	3827	**3823**	**3823**
ran10x26	4270*	**4270**	**4270**	**4270**
ran17x17	1373*	**1373**	**1373**	**1373**

дачах mann_a27, gen200_p0.9_44 и gen200_p0.9_55; $N = 200$ в задаче johnson16.2.4; $N = 400$ в задачах c125.9 и hamming6.4.

На каждой задаче проводится 10 независимых выполнений алгоритма GAgrd с параметрами $s = 20$, $R = 10^7$, $p_{\text{x}} = 0.9$, $N = 10^3$, $p_{\text{cross}} = 0.8$.

Результаты экспериментов. Лучшие решения, найденные описанными выше алгоритмами, представлены в Таблицах 5.8 – 5.12. Здесь f^h_{cplex} соответствует результатам, полученным CPLEX с эвристическими настройками в качестве отдельного модуля. Лучшие значения целевой функции, полученные в наших экспериментах, обозначены жирным шрифтом. Лучшее известное значение целевой функции указано в столбце f_{best} (в случае, если известно о его оптимальности, значение отмечается символом «*»). Если алгоритм закончил работу, не получив допустимого решения задачи, указывается символ «-». Таблицы 5.9 и 5.10 также содержат столбец LB, где указаны нижние оценки, полученные пакетом CPLEX со стандартным набором настраиваемых параметров при том же времени вычислений, что и у других алгоритмов. В сериях RAN, H и MIS значения f_{best} получены из известных решений для тестовых примеров транспортной задачи с фиксированными затратами и задачи о наибольшем независимом множестве.

Лучшие известные решения для серии H получены генетическим алгоритмом GA NLO-1 в [170], который был специально разработан для ре-

Таблица 5.9. Результаты для серии S

	LB	f_{best}	f^h_{cplex}	f_{GAmipr}	f_{GAgrd}
s_1	604685	613951	**613951**	**613951**	614289
s_2	622470	631219	**631219**	**631219**	631540
s_3	617403	627294	**627294**	**627294**	627392
s_4	622761	634609	636855	**634609**	634872
s_5	595627	605145	**605145**	605380	605432
s_6	655595	669346	**669371**	669787	670730
s_7	589851	602737	**602737**	**602737**	603788
s_8	633324	645843	**645843**	**645843**	647242

Таблица 5.10. Результаты для серии H

	LB	f_{best}	f^h_{cplex}	f_{GAmipr}	f_{GAgrd}
h_0	1002063	1208672	1260048	**1217305**	1284070
h_1	990406	1204194	1241812	**1231028**	1303920
h_2	991110	1192687	1269582	**1198609**	1305210
h_3	954486	1188904	1236705	**1215025**	1305520
h_4	992374	1209357	1283998	**1238455**	1332310
h_5	997677	1202258	1264337	**1214291**	1311000
h_6	990739	1201097	1258163	**1235884**	1305000
h_7	967693	1172931	1259481	**1204284**	1281000
h_8	984973	1198108	1272177	**1236018**	1306000
h_9	972743	1192529	1267414	**1227936**	1308000
h_a	994451	1233688	1294036	**1259553**	1304000
h_b	1003666	1208211	1289744	**1237080**	1319000
h_c	990923	1209685	1260134	**1233882**	1338000
h_e	987160	1185378	1257240	**1195558**	1326000

шения транспортных задач с фиксированными затратами. На каждом примере серии H этот алгоритм независимо выполнялся 15 раз, и критерием остановки было построение 10^7 различных пробных решений.

Лучшие известные решения для серии P получены описанными выше алгоритмами. Допустимое решение для задачи p8 удалось получить лишь с помощью гибридного алгоритма, где сначала GAgrd вычисляет некоторое недопустимое решение и затем данное решение используется как начальный рекорд в методе ветвей и отсечений из пакета CPLEX, а в качестве критерия служит функция штрафа.

Таблица 5.11. Результаты для серии P

	LB	f_{best}	f^h_{cplex}	f_{GAmipr}	f_{GAgrd}
p_1	8372	8527	-	**8527**	8597
p_2	7734	7836	**7836**	7840	7865
p_3	8259	8423	-	**8423**	8513
p_4	8086	8409	-	-	**8574**
p_5	7942	8007	8011	**8007**	8027
p_6	8213	8320	8369	**8320**	8392
p_7	7794	7873	**7874**	7876	7889
p_8	8362	9543	-	-	-
p_9	8424	8831	-	-	**8928**
p_10	8360	8524	-	**8524**	8603
p_11	8048	8196	-	**8196**	8263
p_12	8152	8323	-	**8323**	8547
p_13	8036	8276	-	8696	**8276**
p_14	8471	8916	-	-	**8916**

Таблица 5.12. Результаты для серии MIS

	f_{best}	f^h_{cplex}	f_{GAmipr}	f_{GAgrd}
hamming6-4	60*	**60**	**60**	**60**
c125.9	91*	**91**	**91**	94
johnson16.2.4	112*	**112**	**112**	**112**
mann.a27	252*	-	262	**259**
gen200_p0.9_44	156*	166	**164**	171
gen200_p0.9_55	145*	**164**	165	171

При решении задач p_3, gen200_p0.9_44, mann_a27 и gen200_p0.9_55 пакет CPLEX, используемый как отдельный модуль, останавливался из-за нехватки памяти или ошибок округления.

Таблица 5.13 содержит значения средней относительной погрешности алгоритмов, вычисляемые по формуле $r = (f/f_{\text{best}} - 1) \cdot 100\%$, где f – значение целевой функции для решения, полученного указанным алгоритмом. Значения погрешности усреднялись по всем задачам в серии за исключением серии MIS, где задача mann_a27 не учитывается.

Как видно из Таблиц 5.8–5.13, в целом по 50 тестовым примерам результаты алгоритма GAmipr не уступают результатам пакета CPLEX в 44

Таблица 5.13. Средняя относительная погрешность алгоритмов

Series	r^h_{cplex}	r_{GAmipr}	r_{GAgrd}
RAN	0.17 %	0.07 %	0.17 %
S	0.05 %	0.02 %	0.10 %
H	4.80 %	2.00 %	9.06 %
MIS	9.76 %	9.46 %	14.87 %

случаях, а в 28 из этих случаев решение GAmipr оказалось лучше. Средняя относительная погрешность алгоритма GAmipr ниже, чем у двух других алгоритмов на всех сериях.

Алгоритм GAgrd оказался менее точным по сравнению с GAmipr и CPLEX на сериях RAN, S и H, где допустимые решения найти относительно легко. Однако на сериях P и MIS данный алгоритм оказывается наиболее надежным: в 19 из 20 случаев им найдены допустимые решения, в то время как CPLEX за отведенное время оказался неспособен найти допустимое решение в 11 случаях, а GAmipr – в 4 случаях.

Тестирование ЦЛП-кроссинговера на основе ЗОР. Рассмотренные выше эксперименты были проведены повторно с сериями RAN, H, S и P с модифицированным вариантом GAmipr, отличающимся от базового в том, что в операторе ЦЛП-кроссинговера вместо ослабленной ЗОР решается ЗОР согласно определению 3.1. Как выяснилось, данная модификация в целом снижает эффективность ГА при решении исследуемых задач. Только в одной задаче в каждой из серий H и RAN и в двух случаях серии P модифицированным алгоритмом были получены лучшие по целевой функции решения, чем с помощью базового варианта GAmipr, в 19 случаях результаты были одинаковыми, а в 21 случае базовый вариант GAmipr имел преимущество (13 задач в серии H и по 4 задачи в каждой из серий S и P).

Индивидуальные задачи серий H, S и P построены с помощью генераторов псевдослучайных чисел как реализации независимых случайных

величин, которые в каждой серии одинаково распределены. Таким образом, значения целевой функции, полученные в результате применения рандомизированного алгоритма к заданной серии задач, могут рассматриваться как последовательность одинаково распределенных случайных величин. Для сравнения двух вероятностных алгоритмов на основе результатов такого эксперимента уместно использовать непараметрические критерии для парных выборок. В данном случае для проверки гипотезы о равенстве средних значений критерия, полученных базовым вариантом GAmipr и его модификацией, использовался критерий Вилкоксона. Преимущество базового варианта оказалось статистически значимым для серий H (с уровнем значимости 0.0012) и P (с уровнем значимости 0.035), а для серии S уровень значимости различий оказался равен 0.067.

5.2.6. Заключение

Представленные в данном разделе результаты показывают, как интеграция метода ветвей и границ в оператор ЦЛП-кроссинговера позволяет создать конкурентоспособный генетический алгоритм даже в случае, когда соответствующая задача оптимальной рекомбинации является NP-трудной. В ходе вычислительного эксперимента ГА с оператором ЦЛП-кроссинговера показал преимущество в сравнении с алгоритмом ветвей и границ из пакета CPLEX, настроенным на поиск приближенных решений.

Работоспособность предложенного оператора ЦЛП-кроссинговера связана со сравнительно малой размерностью ослабленных задач оптимальной рекомбинации, возникающих при исключении из рассмотрения всех переменных, которые равны нулю в обоих родительских решениях. Возникновение подобного эффекта можно предположить и во многих других задачах частично целочисленного линейного программирования, где векторы допустимых решений содержат большое число нулевых компонент.

Кроме того, в настоящем разделе разработан более простой вариант ГА с декодирующей процедурой, основанной на использовании 2-приближенного жадного алгоритма. Данный алгоритм показал лучшие результаты по количеству тестовых задач, для которых получено допустимое решение. Другое преимущество данного алгоритма состоит в том, что он может быть применен к задаче управления поставками с нелинейными вогнутыми функциями стоимости.

Исследование сложности точного и приближенного решения, проведенное в пп. 5.2.1 – 5.2.2 и в работах [41, 142, 144, 250], показывает, что ввод дополнительного ограничения снизу на допустимый объем партии превращает эффективно разрешимую транспортную задачу в труднорешаемую и во многих случаях неаппроксимируемую задачу. Сложность некоторых задач теории расписаний с аналогичным ограничением снизу на объем партии исследовалась в [162]. Ввиду наличия ограничений снизу на допустимый объем партии во многих практически значимых задачах дальнейшее исследование эволюционных алгоритмов решения таких задач представляет большой интерес. Пример адаптации вычислительной схемы генетического алгоритма с жадной декодирующей эвристикой к практически значимой задаче составления расписаний содержится в [138].

Схема жадного алгоритма 5.4, а также доказательство теоремы 5.2 предложены автором совместно с А.А. Романовой. Генетический алгоритм с использованием жадного декодера (п. 5.2.3) предложен и исследован в эксперименте совместно с П.А. Борисовским и А.Б. Долгим.

5.3. Задача балансировки автоматизированной производственной линии

В данном разделе рассматривается задача балансировки автоматизированной производственной линии и предлагается генетический алго-

ритм с оператором ЦЛП-кроссинговера для ее решения. Исследуемая задача сформулирована в работе [166] и имеет ряд отличий от известной задачи балансировки автоматической сборочной линии [124, 129, 192], таких как параметризованное время выполнения операции, нестрогие отношения предшествования и параллельное выполнение операций. Из-за этих особенностей применение к рассматриваемой задаче известных методов, разработанных для задач балансировки автоматической сборочной линии, становится невозможным [165].

Предложенный ГА сравнивается в вычислительном эксперименте с другими эвристическими и точными алгоритмами. Выявлены классы задач, для которых использование данного ГА является целесообразным. Эксперимент показал перспективность использования задачи оптимальной рекомбинации в ГА для исследуемой задачи.

Рассматривается автоматизированная производственная линия, состоящая из последовательно расположенных синхронно работающих *рабочих станций* (далее для краткости называемых просто *станциями*), соединенных транспортным механизмом. Каждая станция оборудована несколькими *шпиндельными головками* (далее просто *головками*). Каждая головка выполняет свой блок операций по механической обработке детали. Все операции блока выполняются одновременно. Параллельное выполнение операций в блоке возможно благодаря тому, что головки имеют несколько одновременно применяемых инструментов. Головки на каждой станции активируются последовательно.

Когда обработка на текущей станции окончена, т. е. все блоки операций, размещенные на этой станции, выполнены, деталь передвигается к следующей станции. Промежуток времени между переходом детали от одной станции к другой, называемый временем цикла, не должен превышать заданного значения T_0. Переход к следующей рабочей станции происходит одновременно для всех деталей на линии. Необходимо назначить заданное

множество операций на блоки и станции при некоторых ограничениях на допустимые назначения, которые описываются подробно в п. 5.3.1. Требуется минимизировать затраты по сооружению станций и установке головок.

Для решения рассматриваемой задачи разработано несколько точных методов, основанных на ЧЦЛП и методах теории графов, а также эвристические алгоритмы. Краткое описание этих методов содержится в [207]. Позднее в [160, 208, 209] был предложен ряд рандомизированных эвристических алгоритмов для решения этой задачи.

Раздел организован следующим образом. В п. 5.3.1 приводится постановка задачи и модель ЧЦЛП. В п. 5.3.2 описывается эвристика GRASP, а в п. 5.3.3 – ГА с ЦЛП-кроссинговером, аналогичным оператору кроссинговера, предложенному в п. 5.2.4 предыдущего раздела. Результаты вычислительных экспериментов представлены в п. 5.3.4.

5.3.1. Постановка задачи и модель частично целочисленного линейного программирования

Входными данными для задачи балансировки автоматизированной производственной линии являются следующие величины.

- $\mathbf{N}$ – множество всех операций, необходимых для обработки детали.
- T_0 – максимально допустимое время цикла.
- τ^S и τ^b – соответственно, время, необходимое для активации головки, и время, необходимое для загрузки/разгрузки детали на станции.
- C_1 и C_2 – соответственно, относительная стоимость одной станции и относительная стоимость одного блока;
- m_0 – максимальное число станций.
- n_0 – максимальное число блоков на станции;

- *Отношения предшествования* между операциями. Согласно технологическому процессу на множестве операций установлен частичный порядок, заданный с помощью ориентированного графа $G = (\mathbf{N}, D)$. Дуга $(i, j) \in \mathbf{N}^2$ принадлежит D тогда и только тогда, когда блок с операцией j не может предшествовать блоку с операцией i.

- *Отношения включения* определены для группы операций, которые должны быть назначены на одну и ту же машину по технологическим требованиям. Отношения включения задаются семейством ES подмножеств множества $\mathbf{N}$, так что все операции одного подмножества $e \in ES$ должны быть назначены на одну станцию. Никакие два множества семейства ES не могут содержать одну и ту же операцию.

- *Отношения исключения для станции* представляют собой группу операций, которые не могут быть назначены на одну станцию по технологическим требованиям. Задаются семейством $\overline{ES}$ подмножеств из $\mathbf{N}$, так что каждое подмножество $e \in \overline{ES}$ не может быть полностью назначено на одну и ту же станцию.

- *Отношения исключения для блоков* определены на группе операций, которые не могут быть назначены на одну и ту же головку. Задаются семейством $\overline{EB}$ подмножеств из $\mathbf{N}$, так что все операции каждого подмножества $e \in \overline{EB}$ не могут принадлежать одному блоку.

- Возможны два варианта задания времени выполнения операции. Либо для каждой операции j ее длительность t_j указана в явном виде. Либо для каждой операции j заданы максимально допустимая глубина подачи инструмента в минуту s_j и длина рабочего хода инструмента λ_j, включающая в себя глубины выполняемого разреза и расстояния между рабочим инструментом и поверхностью детали.

Сформулированная задача NP-трудна, так как ее частным случаем является NP-трудная задача об одномерной упаковке в контейнеры. В свя-

зи с этим представляет интерес ее изучение в терминах ЦЛП и разработка переборных алгоритмов для поиска оптимального решения.

Модель ЧЦЛП для данной задачи была предложена А.Б. Долгим, Б. Финель, Н.Н. Гущинским и др. в [165]. Рассмотрим модификацию этой модели с усовершенствованием, предложенным О.Н. Гущинской в [207].

Пусть N_k – множество операций, назначенных на станцию k, а N_{kl} – множество операций, сгруппированных в один блок l станции k, $l = 1, \ldots, n_k$, где $n_k = |N_k|$. Обозначим через $t^b(N_{kl})$ время обработки блока l на станции k. Время обработки детали на станции k есть $t^S(N_k) = \sum_{l=1}^{n_k} t^b(N_{kl}) + \tau^S$, так как блоки активируются последовательно.

Рассматриваются два способа задания времени обработки блока. Упрощенное определение [207] основано на предположении о том, что это время равно длительности наиболее трудоемкой операции в блоке:

$$t^b(N_{kl}) = \max\{t_j | j \in N_{kl}\} + \tau^b. \tag{5.40}$$

Более общее определение из [209] не учитывает величин t_j, а основывается на значениях параметров λ_j и s_j:

$$t^b(N_{kl}) = \frac{\max\{\lambda_i | i \in N_{kl}\}}{\min\{s_i | i \in N_{kl}\}} + \tau^b. \tag{5.41}$$

Заметим, что последнее определение действительно обобщает первое, так как достаточно положить $\lambda_j = t_j$, $s_j = 1$ для всех j. Кроме того, общее определение более адекватно для возникающих на практике ситуаций, когда операции, назначенные на несколько инструментов одной головки, могут требовать различной длины рабочего хода при различной максимальной скорости подачи инструмента.

В модели ЦЛП используются следующие обозначения:

- q – индекс для сквозной нумерации всех блоков, например, для блока l станции k имеем $q = (k-1)n_0 + l$;
- q_0 – максимальное возможное значение q, $q_0 = m_0 n_0$;

- $S(k) = \{(k-1)n_0 + 1, \ldots, kn_0\}$ – множество индексов блоков, выполняемых на станции k;
- $Q(j)$ – множество индексов блоков q, в которые может быть назначена операция j;
- $K(j)$ – множество индексов станций k, в которые может быть назначена операция j;
- e – множество операций, являющееся элементом ES, $\overline{ES}$ или $\overline{EB}$;
- $j(e)$ – некоторая зафиксированная операция из множества e;
- t_j – длительность операции j: в упрощенном варианте t_j задается как часть входных данных, иначе $t_j = \frac{\lambda_j}{s_j}$;
- $t_{ij} = \frac{\max\{\lambda_i, \lambda_j\}}{\min\{s_i, s_j\}}$ – время выполнения двух операций i, j, если они выполняются в одном блоке (это значение не используется в упрощенном варианте постановки или если величины s_j все одинаковы).

Переменные:

- x_{jq} – бинарная переменная, принимающая значение 1, если операция j назначена в блок q, иначе ее значение равно 0;
- f_q – вещественная переменная, служащая для подсчета времени выполнения блока q;
- y_q – бинарная переменная, принимающая значение 1, если блок q существует, иначе ее значение равно 0;
- z_k – бинарная переменная, принимающая значение 1, если станция k существует, иначе ее значение равно 0.

Переменные y_q и z_k необходимы для подсчета числа блоков и станций соответственно. Анализ ограничений данной задачи, проведенный в [165],

позволил установить диапазоны возможных значений индексов во множествах $Q(j)$ и $K(j)$ для каждой операции j, что дало существенное сокращение числа переменных и ограничений в модели.

Задача балансировки автоматизированной производственной линии формулируется в виде задачи ЧЦЛП следующим образом. Найти

$$\min\ C_1 \sum_{k=1}^{m_0} z_k + C_2 \sum_{q=1}^{q_0} y_q \tag{5.42}$$

при условиях

$$\sum_{q\in Q(j)} (q-1)x_{jq} \geq \sum_{s\in Q(i)} (s-1)x_{is}, \quad (i,j)\in D, \tag{5.43}$$

$$\sum_{q\in Q(j)} x_{jq} = 1, \quad j\in \mathbf{N}, \tag{5.44}$$

$$\sum_{j\in e\setminus\{j(e)\}} \sum_{q\in S(k)\cap Q(j)} x_{jq} = (|e|-1) \sum_{q\in S(k)} x_{j(e)q}, \quad e\in ES,\ k\in K(j(e)), \tag{5.45}$$

$$\sum_{j\in e} x_{jq} \leq |e|-1, \quad e\in \overline{EB},\ q\in \cap_{j\in e} Q(j), \tag{5.46}$$

$$\sum_{j\in e} \sum_{q\in S(k)\cap Q(j)} x_{jq} \leq |e|-1, \quad e\in \overline{ES},\ k\in \cap_{j\in e} K(j), \tag{5.47}$$

$$f_q \geq (t_i+\tau^b)x_{iq}, \quad i\in \mathbf{N},\ q\in Q(i), \tag{5.48}$$

$$f_q \geq (t_{ij}+\tau^b)(x_{iq}+x_{jq}-1), \quad i,j\in \mathbf{N},\ i<j,\ q\in Q(i)\cap Q(j), \tag{5.49}$$

$$\tau^S + \sum_{q\in S(k)} f_q \leq T_0, \quad k=1,2,\ldots,m_0, \tag{5.50}$$

$$y_q \geq x_{jq}, \quad j\in \mathbf{N},\ q\in Q(j), \tag{5.51}$$

$$z_k = y_{(k-1)n_0+1}, \quad k = 1, 2, \ldots, m_0, \tag{5.52}$$

$$y_{q-1} - y_q \geq 0, \quad q \in S(k) \backslash \{(k-1)n_0 + 1\}, \; k = 1, 2, \ldots, m_0, \tag{5.53}$$

$$z_{k-1} - z_k \geq 0, \quad k = 2, 3, \ldots, m_0, \tag{5.54}$$

$$x_{jq}, \; y_q, \; z_k, \in \{0, 1\}, \quad j \in \mathbf{N}, \; q = 1, 2, \ldots, q_0, \; k = 1, \ldots, m_0, \tag{5.55}$$

$$f_q \in [0, T_0 - \tau^S - \tau^b], \quad q = 1, 2, \ldots, q_0. \tag{5.56}$$

Ограничения предшествования задаются с помощью неравенств (5.43). Ограничения (5.44) гарантируют, что каждая операция должна быть назначена только в один блок. Условия (5.45) определяют необходимость группировки соответствующих операций в одну станцию. Ограничения (5.46) и (5.47) запрещают группировку соответствующих операций в один блок и выполнение определенных операций на одной станции соответственно. Ограничения (5.48) и (5.49) определяют время выполнения блока: неравенства (5.48) соответствуют случаю с одной операцией в блоке, а неравенства (5.49) – случаям с двумя и более операциями. Заметим, что в случае упрощенной формулировки, где длительность выполнения блока задается условием (5.40), или если все величины s_j одинаковы, неравенства (5.49) являются избыточными. Ограничения (5.50) определяют время цикла. Условия (5.51) обеспечивают наличие блока q в решении тогда и только тогда, когда $x_{jq} = 1$ по крайней мере для одной операции j. Условия (5.52) обеспечивают наличие станции k в решении тогда и только тогда, когда $y_q = 1$ по крайней мере для одного блока $q \in S(k)$. Ограничения (5.53) гарантируют, что блок q

на станции k может быть создан только если блок $q-1$ существует на этой станции. Ограничения (5.54) обеспечивают использование станции k только если в линии существует станция $k-1$.

Неравенства (5.53) и (5.54) в этой модели введены главным образом как нарушающие симметрию отсечения (нетрудно заметить, что простой модификацией условий (5.52) эти неравенства можно сделать избыточными). Граничные условия (5.55) также введены с целью сокращения многогранника релаксированной задачи.

Как показали эксперименты, улучшение работоспособности алгоритмов ветвей и границ с отсечениями может быть получено при добавлении некоторого малого штрафа к целевой функции, дающего при прочих равных условиях небольшое преимущество тем решениям, где операции назначены в блоки с меньшими порядковыми номерами. Чем выше порядковый номер блока, в который назначена операция, тем больше штраф:

$$\min C_1 \sum_{k=1}^{m_0} z_k + C_2 \sum_{q=1}^{q_0} y_q + C_3 \sum_{q=1}^{q_0} \sum_{j \in \mathbf{N}} (1-q) x_{jq}. \qquad (5.57)$$

Здесь вес C_3 выбирается достаточно малым, чтобы оптимальное решение модифицированной задачи (5.43)–(5.57) было бы также оптимальным для задачи (5.42)–(5.56).

5.3.2. Эвристика GRASP

Эвристика GRASP представляет собой итерационный метод, каждая итерация которого состоит из двух этапов: построение допустимого решения и улучшение имеющегося решения методом локального поиска. Итерации продолжаются до тех пор, пока не будет выполнен некоторый критерий остановки, и результатом работы является лучшее из найденных решений. В процессе построения допустимого решения используется некоторый вариант жадного алгоритма с рандомизацией (см., например, [210]). На подобных идеях основаны алгоритмы случайного поиска с адаптацией [76]

и алгоритмы муравьиных колоний (см., например, [111]). Аббревиатура GRASP (Greedy Adaptive Search Procedure) предложена в [183]. Обзоры по применению GRASP содержатся в [184, 274].

Построение допустимого решения

В случае задачи балансировки автоматизированной производственной линии допустимое решение также может быть построено рандомизированным жадным алгоритмом [208]. Рассмотрим аналогичный алгоритм, основанный на решении серии подзадач ЧЦЛП вида (5.42)–(5.56). Построение решения начинается с линии, в которой ни один блок не содержит операций. На итерации t' размещается некоторое подмножество операций $N^{t'}_{\text{доб}}$. Обозначим через $N^{t'}$ множество операций, которые уже были размещены на шагах $1, \ldots, t'$, предполагая, что $N^0 = \emptyset$.

Подмножество $N^{t'}_{\text{доб}}$ вычисляется рандомизированной процедурой с настраиваемыми параметрами $\alpha \in [0, 1]$ и $\beta \in \{1, \ldots, |\mathbf{N}|\}$ (выбор значений α и β обсуждается в [160]). На шаге t' сначала полагаем $N^{t'}_{\text{доб}} = \emptyset$, затем расширяем это множество с помощью следующей циклической процедуры.

1. Вычислить *множество операций-кандидатов* N_{CL}, состоящее из операций, которые могут выполняться, если завершены все операции из множества $N^{t'-1} \cup N^{t'}_{\text{доб}}$.

2. Упорядочить все операции из множества N_{CL} по значениям жадного критерия $g(j)$ (эта функция будет определена ниже). Найти

$$g_{max} = \max\{g(j) : j \in N_{CL}\} \text{ и } g_{min} = \min\{g(j) : j \in N_{CL}\}.$$

3. Поместить все те операции-кандидаты j, для которых выполнено неравенство $g(j) \geq g_{max} - \alpha(g_{max} - g_{min})$, во множество N_{RCL}, называемое *сокращенным списком операций-кандидатов.*

4.Выбрать элемент j равновероятно из множества N_{RCL}. Добавить j во множество $N^{t'}_{\text{доб}}$.

5. Включить в $N^{t'}_{\text{доб}}$ все операции i, такие что $(i,j) \in D'$ и $(j,i) \in D'$.

Перечисленные действия выполняются в цикле β раз. Цикл завершается раньше, если не останется ни одной операции для добавления, т. е. $N^{t'-1} \cup N^{t'}_{\text{доб}} = \mathbf{N}$. Итерации жадного алгоритма продолжаются до тех пор, пока не выполнится хотя бы одно из двух условий: либо все операции были назначены и получено допустимое решение, либо новую станцию невозможно создать ввиду того, что $m+1 > m_0$.

Вспомогательный жадный критерий $g(j)$ позволяет оценить эффект от назначения операции j на текущей итерации. Значение функции $g(j)$ представляет собой нижнюю оценку числа блоков, требуемых для назначения всех операций, следующих за операцией j. Для вычисления этой оценки используется алгоритм, предложенный в [165].

После выбора множества $N^{t'}_{\text{доб}}$ операции этого множества включаются в текущее частичное решение, вычисленное на предыдущих итерациях. Для всех j и q, $j \in \mathbf{N}$, $q = 1, \ldots, q_0$, определим набор величин $x^{(t')}_{jq}$ следующим образом: $x^{(t')}_{jq} = 1$, если операция j назначена в блок q текущего решения на итерации t'; $x^{(t')}_{jq} = 0$ иначе. Назначение выбранных операций множества $N^{t'}_{\text{доб}}$ может быть осуществлено с помощью вспомогательной задачи ЧЦЛП с использованием подхода, предложенного в [14, 138]. Множество ограничений задается условиями (5.43)–(5.56) при целевой функции (5.57), однако, в отличие от исходной задачи, большое число булевых переменных полагаются равными нулю. Назначение нулевых значений переменным осуществляется следующим образом.

Пусть $k_{\text{исп}}$ обозначает номер последней станции, на которую размещены какие-либо операции в текущем частичном решении, т. е.

$$k_{\text{исп}} = \max\{k \mid \sum_{j \in \mathbf{N}} \sum_{q \in S(k)} x^{(t'-1)}_{jq} \geq 1\}.$$

Потребуем, чтобы операции множества $N^{t'}_{\text{доб}}$ размещались на станции с номерами не более $k_{\max} = k_{\text{исп}} + \beta$ (такое частичное решение всегда

можно найти, если исходная задача разрешима и $k_{\max} \leq m_0$). С этой целью оставим свободными переменные x_{jq} при $j \in N^{t'}_{\text{доб}}$ и $q \leq q_{max}$, где $q_{max} = k_{\max} \cdot n_0$. Также оставим свободными переменные x_{jq}, такие что $x_{jq}^{(t'-1)} = 1$ или $q = 1 + \max\{q | \sum_{j \in \mathbf{N}} x_{jq}^{(t'-1)} \geq 1\}$, для того, чтобы позволить перемещение некоторых ранее размещенных операций в первый из блоков, создаваемых на данной итерации (такое перемещение может снижать стоимость решения). Прочие переменные x_{jq} полагаются равными 0.

Получаемая задача ЧЦЛП на каждом шаге t', вообще говоря, является NP-трудной (например, при $t' = 1$ и $\beta = m_0$ эта задача эквивалентна исходной). Для решения данной подзадачи может использоваться метод ветвей и границ. С увеличением параметра β, как правило, погрешность жадного алгоритма снижается, поэтому значение β выбирается экспериментально таким образом, чтобы обеспечить баланс между трудоемкостью решения вспомогательных задач и точностью получаемых решений.

Этап улучшения решения

Для улучшения допустимого решения предлагается использовать модификацию декомпозиционного алгоритма DAASS [208].

Декомпозиция исходной задачи состоит в разбиении последовательности станций, соответствующей начальному допустимому решению, на несколько непересекающихся подпоследновательностей. Далее проводится оптимизация части решения, соответствующей каждой подпоследовательности в порядке прохождения линии. Длина каждой подпоследовательности выбирается случайным образом, как будет описано ниже. Общее число w таких подпоследовательностей определяется только по окончании процедуры декомпозиции.

Введем обозначение $K_r,\ r = 1, \ldots, w$, для подпоследовательности, содержащей случайное число станций k_r, а соответствующую такой подпоследовательности вспомогательную задачу обозначим через SP_r. При по-

строении случайных подмножеств K_r используются следующие параметры, ограничивающие размеры вспомогательных задач [208]: (i) максимальное число станций $k_{\max}$ в одной подпоследовательности; (ii) максимальное число операций $n_{\max}$ в одной подпоследовательности.

Значение k_r, $r = 1, \ldots, w$, выбирается равновероятно из $\{1, \ldots, k_{\max}\}$ и впоследствии может быть скорректировано так, чтобы число операций во вспомогательной задаче SP_r не превышало порога N_{max}, а сумма $\sum_{r=1}^{w} k_r$ – общего числа станций в текущем решении.

Вспомогательные задачи SP_r, $r = 1, \ldots, w$, формулируются как задачи ЧЦЛП в постановке (5.43)–(5.57). Все булевы переменные, не относящиеся к станциям множества K_r, полагаются равными константам – их значения выбираются в соответствии с наилучшим полученным решением. Остальные переменные $x_{jq},\ y_q,\ q \in \cup_{k \in K_r} S(k)$ и $z_k, k \in K_r$ подлежат оптимизации. Для решения задач SP_r используются методы ветвей и границ.

В отличие от описанного алгоритма улучшения решений в алгоритме DAASS [208] при формулировке вспомогательных задач применяется агрегация блоков операций. Каждый блок операций, полученный в результате решения текущей вспомогательной задачи, в последующем заменяется одной *макрооперацией*, и все макрооперации также подлежат оптимизации при решении следующей вспомогательной задачи. Для решения вспомогательных задач в DAASS используется алгоритм отыскания кратчайшего пути во вспомогательном графе специального вида [167].

5.3.3. Генетический алгоритм с оптимальной рекомбинацией

В предлагаемом ГА применяется стационарная стратегия управления популяцией и турнирная селекция. Функция $\Phi(\xi)$ является строго убывающей относительно стоимости решения, кодируемого генотипом ξ.

Каждый раз после выполнения условия остановки данный ГА возобновляет работу с новой начальной популяции, построенной независимо от

предыдущих. Такой процесс вычислений продолжается до тех пор, пока не будет выполнен некоторый «внешний» критерий остановки. В ходе вычислительных экспериментов в п. 5.3.4 таким «внешним» критерием остановки является общее время вычислений T. Лучшее из найденных решений за время работы алгоритма рассматривается как окончательный результат.

Пусть θ – номер итерации, когда имело место последнее улучшение рекорда по целевой функции. Условие остановки ГА формулируется следующим образом: завершить цикл, если в течение последних θ итераций рекорд по целевой функции не был улучшен и, кроме того, $t > N$.

В качестве генотипа в данном ГА используется совокупность булевых переменных (x_{jq}), которая полностью описывает назначение операций на блоки и станции. Генотипы начальной популяции формируются случайным образом N-кратным применением эвристики GRASP (см. п. 5.3.2).

ЦЛП-кроссинговер. Аналогично оператору ЦЛП-кроссинговера, предложенному в разделе 5.2, оператор ЦЛП-кроссинговера для рассматриваемого ГА сочетает в себе функции мутации и рекомбинации. В данном случае действие оператора ЦЛП-кроссинговера состоит в решении вспомогательной задачи ЧЦЛП, получаемой из исходной задачи (5.42)–(5.56) следующим образом. Пусть задано два родительских генотипа $\mathbf{p}^1 = (p^1_{jq}), \mathbf{p}^2 = (p^2_{jq})$. Добавим к ограничениям исходной задачи условие «фиксации» переменных: $x_{jq} = p^1_{jq}$ при всех j, q, таких что $p^1_{jq} = p^2_{jq}$, кроме некоторого случайного подмножества переменных S_{mut}:

$$x_{jq} = p^1_{jq} \text{ для всех } j, q : \; p^1_{jq} = p^2_{jq} \text{ и } (j,q) \notin S_{\text{mut}}. \tag{5.58}$$

Всякая пара индексов $(j,q) : p^1_{jq} = p^2_{jq}$ независимо от других таких пар, включается в S_{mut} с заданной вероятностью P_{m}.

В частном случае при $P_{\text{m}} = 0$ вспомогательная задача ЦЛП-кроссинговера представляет собой задачу оптимальной рекомбинации. Задача балансировки автоматизированной производственной линии в каче-

стве частного случая содержит задачу об одномерной упаковке в контейнеры (см. п. 3.3.2). Поэтому, ввиду теоремы 3.3, ЗОР для задачи балансировки автоматизированной производственной линии NP-трудна.

Если найдено оптимальное или некоторое приближенное решение вспомогательной задачи, то набор значений (x_{jq}) из этого решения становится результатом оператора ЦЛП-кроссинговера.

При реализации ГА решение вспомогательной задачи в операторе ЦЛП-кроссинговера осуществляется с помощью алгоритма ветвей и границ с отсечениями из пакета CPLEX. В качестве начального рекордного решения используется одно из родительских решений. Во избежание чрезмерных вычислительных затрат при выполнении ЦЛП-кроссинговера устанавливается ограничение T_{rec} на время решения вспомогательной задачи.

На начальной итерации вероятность P_{m} полагается равной значению настраиваемого параметра P_{m}^0 и адаптивно модифицируется в процессе работы ГА. Всякий раз, когда вспомогательная задача в ЦЛП-кроссинговере решается до получения оптимального решения, значение P_{m} увеличивается на 10%. Если же за отведенное время T_{rec} оптимум вспомогательной задачи найти не удается и при этом $P_{\mathrm{m}} > P_{\mathrm{m}}^0$, то значение P_{m} снижается на 10%. Описанный способ выбора вероятности мутации позволяет получать подзадачи такой размерности, при которой алгоритм ветвей и границ находит их оптимум за время, близкое к T_{rec}.

5.3.4. Экспериментальное исследование алгоритмов

Предложенный ГА, обозначаемый ниже через GA, сравнивался с эвристикой GRASP и следующими эвристическими и точными алгоритмами:

- мультистартовый гибридный алгоритм декомпозиции (далее обозначаемый HD), сочетающий в себе алгоритм улучшения решений DAASS [207, 208] с эвристикой FSIC [164];

- точный алгоритм решения, основанный на поиске кратчайшего пути в графе специальной структуры [167, 207], обозначаемый далее SP;
- метод ветвей и границ из пакета CPLEX 11.1, применяемый к задаче в постановке (5.42)–(5.56), далее обозначается через CPLEX.

В экспериментах использовались 7 серий тестовых примеров. Серии S1–S5 из работы [207] содержат по 50 примеров, сгенерированных с использованием датчика псевдослучайных чисел. Серии S6 и S7 содержат по 20 примеров, которые также сгенерированы случайным образом, и наиболее приближены по своим параметрам к задачам, возникающим на практике.

Исходные данные отличаются по числу операций, необходимых для обработки детали, числу подмножеств в семействах ES и $\overline{ES}$, по верхней границе на количество создаваемых станций m_0 и по *интенсивности порядка*, заданного графом отношений предшествования. Здесь под интенсивностью порядка (order strength), согласно [282], понимается число ребер в транзитивном замыкании графа отношений предшествования G, деленное на $|\mathbf{N}|(|\mathbf{N}|-1)/2$. Длительности обработки – случайные величины с равномерным распределением на отрезке [10, 20]. Граф отношений предшествования строится начиная с пустого множества дуг. Далее дуги добавляются между случайно выбранными вершинами, пока заданная интенсивность порядка не будет достигнута. Отношения включения и исключения задаются случайным образом, но так, чтобы не противоречить друг другу и гарантировать существование допустимого решения. Значения параметров $|\mathbf{N}|, m_0$ и интенсивность порядка (OS) приведены в Таблице 5.14. Серии S1–S5 имеют параметры $C_1 = 10, C_2 = 2$, $\tau^b = \tau^S = 0$, $n_0 = 4$, а серии S6 и S7 – параметры $C_1 = 1, C_2 = 0.5$, $\tau^b = 0.2, \tau^S = 0.4$, $n_0 = 4$. Описание генератора задач приводится в [207]. Исходные данные тестовых примеров доступны по адресу *http://math.nsc.ru/AP/benchmarks/*.

Вычислительный эксперимент проводился на ЭВМ с ЦПУ Pentium-IV 3 ГГц и 2.5 Гб ОЗУ. Как GA, так и GRASP были реализованы в системе

Таблица 5.14. Тестовые серии

Серия	1	2	3	4	5	6	7
$\|\mathbf{N}\|$	25	25	50	50	100	46 - 92	94 - 125
m_0	15	4	10	15	15	23 - 46	43 - 62
OS	0.5	0.15	0.9	0.45	0.25	-	-

GAMS 22.8, прочие алгоритмы реализованы на Visual C++ 6.0.

На основе предварительных экспериментов [160] были выбраны следующие значения параметров эвристики GRASP: $\alpha = 0.25$, $\beta = 10$. Аналогичным образом были выбраны параметры декомпозиционной эвристики улучшения решения в алгоритме GRASP: $k_{\max} = 15$, $n_{\max} = 50$.

Значение параметра C_3 изначально выбирается достаточно малым так, чтобы оптимум модифицированной задачи оставался оптимальным решением и для исходной. Далее значение данного параметра настраивается с помощью простой процедуры одномерного поиска в процессе работы алгоритма GRASP и при построении начальной популяции ГА.

В экспериментах с ГА размер турнира $s = 5$, численность популяции $N = 20$, а начальная вероятность мутации $P_{\mathrm{m}}^0 = 0.1$. Время, отведенное на поиск решений в операторе ЦЛП-кроссинговера, составляет $T_{\mathrm{rec}} = 5$ с.

Экспериментальные результаты

При описании результатов экспериментов используются следующие обозначения. Пусть NS – число примеров, для которых удалось найти допустимое решение; NO и NB – число примеров, для которых, соответственно, оптимальное или лучшее известное решение было найдено; Δ_{max}, Δ_{avg} и Δ_{min} – соответственно, максимальное, среднее и минимальное отклонение (в процентах) полученного решения относительно оптимального или лучшего известного решения; T_{max}, T_{av}, T_{min} – соответственно, максимальное, среднее и минимальное время вычислений. T'_{av} – среднее время получения окончательного решения. Символ «$-$» обозначает отсутствие данных.

Таблица 5.15. Результаты для серий S1 и S2

	S1				S2			
	SP	CPLEX	HD	GRASP	SP	CPLEX	HD	GRASP
NS	50	50	50	50	50	50	50	50
NO	**50**	**50**	39	49	**50**	**50**	35	**50**
Δ_{max}	**0**	**0**	5.26	4.16	**0**	**0**	11.1	**0**
Δ_{av}	**0**	**0**	1.0	0.08	**0**	**0**	1.7	**0**
Δ_{min}	**0**	**0**	0.0	0.0	**0**	**0**	**0**	**0**
T_{max}	**11**	841	90	90	1638	**3.53**	90	90
T_{av}	**1.4**	38	90	90	292	**0.86**	90	90
T_{min}	**0.03**	0.39	90	90	3.77	0.06	90	90

Таблица 5.16. Результаты для серии S3

	SP	CPLEX	HD	GRASP	GA
NS	50	50	50	50	50
NO	50	50	28	48	49
Δ_{max}	**0**	**0**	4.2	2.27	1.72
Δ_{av}	**0**	**0**	1.0	0.09	0.03
Δ_{min}	**0**	**0**	0	0	0
T_{max}	**0.09**	463.4	300	300	300
T_{av}	**0.04**	41.1	300	300	300
T_{min}	**0.01**	0.1	300	300	300

Задачи малой размерности. На сериях S1 и S2 время вычислений ограничивалось сверху 1800 с для точных алгоритмов и 90 с для эвристик. Результаты этих экспериментов представлены в Таблице 5.15.

В сериях S1 и S2 точные алгоритмы за отведенное время обнаружили оптимум в каждой задаче. При решении задач серии S1 алгоритм SP показал лучшие результаты по времени счета, что связано с понижением производительности метода ветвей и границ пакета CPLEX по мере увеличения интенсивности порядка (см., например, [207]). По той же причине точность GRASP при решении задач серии S1 ниже, чем в случае серии S2.

По времени счета в серии S2 лучшие результаты показывает CPLEX. Эвристикой GRASP в этой серии были найдены оптимумы всех задач, в отличие от алгоритма HD. Однако этим двум эвристикам было выделено большее время, чем требовалось пакету CPLEX для поиска оптимума.

Таблица 5.17. Результаты для серии S4

	SP	CPLEX	HD	GRASP	GA
NS	14	20	**50**	**50**	**50**
NO	14	13	39	42	**48**
Δ_{max}	-	-	13.15	13.15	**2.7**
Δ_{av}	-	-	0.86	0.64	**0.11**
Δ_{min}	-	-	**0**	**0**	**0**
T_{max}	1800	1800	300	300	300
T_{av}	1490.3	1519.0	300	300	300
T_{min}	13.0	31.5	300	300	300

Задачи средней размерности. При решении задач средней размерности из серий S3 и S4 (см. Таблицы 5.16 и 5.17) время вычислений ограничивалось 1800 с для точных алгоритмов и 300 с для эвристик.

При решении серии S3 алгоритм, основанный на поиске кратчайшего пути, показывает лучшие результаты как по времени счета, так и по качеству полученных решений. Пакетом CPLEX найдены оптимумы во всех задачах, а алгоритму GRASP не удалось найти оптимум в двух случаях. Среднее и максимальное время получения окончательного решения для GRASP составило 5.02 с и 66 с соответственно. GA продемонстрировал аналогичные результаты, несколько превосходящие результаты GRASP. Алгоритм HD в этих сериях уступает другим по качеству полученных решений.

При решении серии S4 точные алгоритмы обнаружили оптимум менее чем в половине случаев – см. Таблицу 5.17. При этом время вычислений CPLEX и HD было близким. Существенно лучшие результаты в данном случае показали эвристики, несмотря на то, что для них ограничение на время вычислений было в 6 раз меньше, чем для точных методов. В целом, по качеству решений лучшие результаты показал GA.

Задачи большой размерности. Результаты для серии S5 представлены в Таблице 5.18. Время вычислений ограничивалось 5400 с для точных методов и 600 с для эвристик. Точные алгоритмы получили допустимые решения менее чем в 10 случаях, поэтому их результаты не приводятся в

Таблица 5.18. Результаты для серий S5, S6 и S7

	S5			S6			S7		
	HD	GRASP	GA	HD	GRASP	GA	HD	GRASP	GA
NB	11	**37**	36	10	8	**15**	6	2	**17**
Δ_{max}	35.9	30.8	**12.8**	7.6	5.6	**4.4**	6	5.3	**1.3**
Δ_{av}	5.0	1.6	**1.5**	1.8	1.6	**0.7**	2.3	1.9	**0.2**
T'_{av}	-	**93.4**	102.9	-	360.4	423.6	-	1603	1323.5

этой таблице. Эвристикам удалось получить допустимые решения во всех 50 случаях. В данной серии GRASP и GA показывают аналогичные результаты, в то время как HD существенно им уступает.

Таблица 5.18 содержит также результаты для серий S6 и S7, где используется формулировка задачи, основанная на уравнении (5.41). Время вычислений, выделенное для эвристик, составляло 900 с в серии S6 и 3000 с в серии S7. При решении задач из этих серий HD и GRASP показывают близкие результаты. Генетический алгоритм имеет тенденцию к получению лучших результатов, чем две другие эвристики в сериях S6 и S7.

Для оценки значимости «вклада» ЦЛП-кроссинговера были проведены эксперименты, где вероятность выполнения данного оператора задавалась параметром P_r. Когда ЦЛП-кроссинговер не использовался (с вероятностью $1 - P_r$), выполнялась только процедура мутации. На Рисунке 5.5 приведена частота обнаружения решений с лучшим известным значением критерия в зависимости от P_r. За исключением наиболее простой серии S3 увеличение вероятности применения ЦЛП-кроссинговера ведет к увеличению частоты получения решений с лучшим известным значением критерия. Значимость различий подтверждается критерием Вилкоксона с уровнем значимости $p < 0.1$ для S5 и $p < 0.05$ для S6.

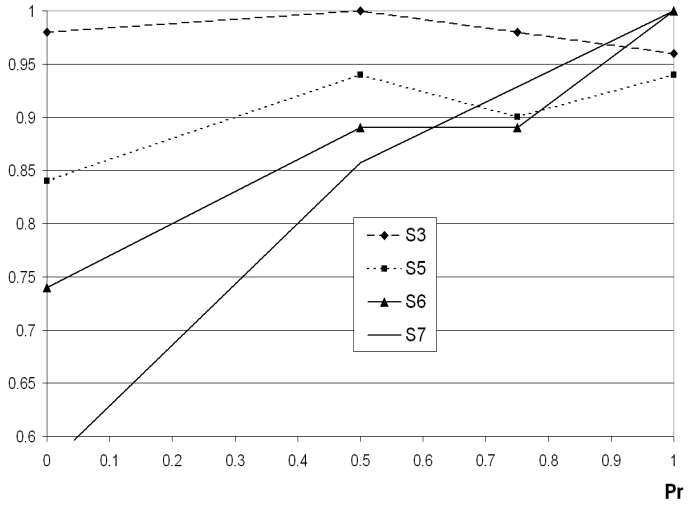

Рис. 5.5. Частота обнаружения решений с лучшим известным значением критерия в зависимости от параметра P_r

5.3.5. Заключение

Вычислительный эксперимент показывает, что предложенный алгоритм ЦЛП-кроссинговера позволяет создать достаточно конкурентоспособный ГА, в особенности на задачах большой размерности.

В ходе дальнейших исследований [209] нами установлено, что использование метода [167] позволяет получать лучшие результаты при решении тестовых примеров, сформулированных с использованием уравнения (5.41) или с высокой интенсивностью порядка. В целом же ГА с ЦЛП-кроссинговером остается одним из наиболее перспективных эвристических методов решения задачи балансировки автоматизированной производственной линии. Важной особенностью данного подхода является возможность его применения к широкому классу задач за счет использования гибких средств построения моделей ЧЦЛП и универсальных пакетов программ для решения такого рода задач. Использование данного подхода особенно актуально в тех случаях, когда непосредственное применение универсальных пакетов целочисленного программирования требует чрезмерных вычислительных затрат.

Заключение

На защиту выносятся следующие основные результаты.

1. Проведено комплексное исследование эволюционных методов с позиций локальной оптимизации: найдены необходимые и достаточные условия, при которых рандомизированный вариант локального поиска является наилучшим методом по вероятности отыскания оптимума в классе эволюционных алгоритмов с заданным оператором мутации; найдены достаточные условия, при которых генетический алгоритм с турнирной селекцией достигает локальный оптимум в среднем за полиномиально ограниченное время, и показано выполнение этих условий на классе задач GLO; предложены статистические методы нахождения нижних оценок числа локальных оптимумов задачи.

2. Исследована сложность задач оптимальной рекомбинации: установлена полиномиальная разрешимость задач оптимальной рекомбинации в генетических алгоритмах для ряда задач булевого линейного программирования, в частности задач упаковки и разбиения множества и простейшей задачи размещения производства; установлена NP-трудность оптимальной рекомбинации для задачи коммивояжера, задач о рюкзаке, кратчайшем гамильтоновом пути, упаковке в контейнеры и некоторых других классических задач комбинаторной оптимизации.

3. Построены эволюционные алгоритмы, аналогичные по свойствам алгоритмам динамического программирования: получены полиномиальные оценки средней трудоемкости многокритериального эволюционного алгоритма через трудоемкость алгоритма динамического программирования; для класса задач, удовлетворяющих условиям существования вполне полиномиальной аппроксимационной схемы Г. Воегингера, предложена вполне

полиномиальная рандомизированная аппроксимационная схема, основанная на эволюционном алгоритме.

4. Разработан подход к построению гибридных генетических алгоритмов, использующих оптимальную рекомбинацию и жадные алгоритмы для решения NP-трудных задач комбинаторной оптимизации: предложены новые генетические алгоритмы для задач покрытия множества, управления поставками продукции и балансировки производственной линии, экспериментально установлены классы задач, на которых такие алгоритмы имеют преимущество по сравнению с известными алгоритмами; выполнен теоретический анализ сложности приближенного решения задачи управления поставками продукции.

5. Проведено исследование генетического алгоритма с турнирной селекцией и мутацией: предложена математическая модель эволюционного процесса в популяции генетического алгоритма; с использованием данной модели построены оценки динамики численности особей с высокой приспособленностью и изучены частные случаи достижимости оценок.

Литература

1. Агеев А. А. О минимизации квадратичных полиномов от булевых переменных // Управляемые системы: сб. науч. тр. – Новосибирск: Ин-т математики СО АН СССР, 1984. – Вып. 25. – С. 3–16.

2. Агеев А. А., Бабурин А. Е., Гимади Э. Х. Полиномиальный алгоритм с оценкой точности 3/4 для отыскания двух непересекающихся гамильтоновых циклов максимального веса // Дискрет. анализ и исслед. операций. Сер. 1. – 2006. – Т.13. – № 2. – С. 11–20.

3. Айзерман М. А., Алескеров Ф. Т. Выбор вариантов: основы теории. – М.: Наука, 1990. – 240 с.

4. Алтухов Ю. П. Генетические процессы в популяциях. – М.: Академкнига, 2003. – 431 с.

5. Антамошкин А. Н., Масич И. С. Гриди-алгоритмы и локальный поиск для условной псевдобулевой оптимизации. Электронный журнал «Исследовано в России», 177, 2143-2149, 2003. http://zhurnal.ape.relarn.ru/articles/1998/003.pdf

6. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536 с.

7. Батищев Д. И., Старостин Н. В. K-разбиение графов // Математическое моделирование и оптимальное управление. Вестник Нижегородского госуниверситета. – Н.Новгород: Изд-во ННГУ, 2000. – С. 25–37.

8. Батищев Д. И., Старостин Н. В., Филимонов А. В. Многоуровневый генетический алгоритм решения задачи декомпозиции гиперграфа //

Информатика, управление и компьютерные технологии. Изв. СПГЭТУ «ЛЭТИ». – СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2007. – Вып. 2. – С. 3–13.

9. Бахтин А. Е., Колоколов А. А., Коробкова З. В. Дискретные задачи производственно-транспортного типа. – Новосибирск: Наука, 1978. – 167 с.

10. Береснев В. Л. Алгоритм неявного перебора для задачи типа размещения и стандартизации // Управляемые системы: сб. науч. тр. – Новосибирск: Ин-т математики СО АН СССР, 1974. – Вып. 12. – С. 24–34.

11. Береснев В. Л. Дискретные задачи размещения и полиномы от булевых переменных. – Новосибирск: Изд-во Ин-та математики, 2005. – 408 с.

12. Береснев В. Л., Гимади Э. Х., Дементьев В. Т. Экстремальные задачи стандартизации. – Новосибирск: Наука, 1978. – 385 с.

13. Борисовский П. А. Генетические алгоритмы для задачи о поставках продукции // Материалы V международной научно-технической конференции «Динамика систем, механизмов и машин». – Омск: Изд-во ОмГТУ, 2004. – Кн. 2. – С. 255–258.

14. Борисовский П. А., Еремеев А. В. Приближенные алгоритмы построения расписаний в многопродуктовом производстве // Тезисы докладов XIII Всероссийской конференции «Математическое программирование и приложения». – Екатеринбург: УрО РАН, 2007. – С. 98. – (Информ. бюл. Ассоциации математ. программирования; № 11).

15. Борисовский П. А., Еремеев А. В. Генетический алгоритм для задачи о вершинном покрытии графа // Математика и информатика: наука и образование. – Омск: ОмГПУ, 2008. – Вып. 7. – С. 49–54.

16. Борисовский П. А., Еремеев А. В. О сравнении некоторых эволюционных алгоритмов // Автомат. и телемех. – 2004. – № 3. – С. 3–10.

17. Боровков А. А. Теория вероятностей. – М.: Наука, 1986. – 432 с.

18. Букатова И. Л., Михасев Ю. И., Шаров А. М. Эвоинформатика: теория и практика эволюционного моделирования. – М.: Наука, 1991. – 206 с.

19. Булатов В. П. Методы погружения в задачах оптимизации. – Новосибирск: Наука, 1977. – 161 с.

20. Быкова В. В. Проблема памяти в динамическом программировании // V Всероссийская конференция «Проблемы оптимизации и экономические приложения»: материалы конференции. – Омск: Изд-во Омского гос. ун-та, 2012. – С. 6.

21. Воегингер Г. Д., Севастьянов С. В. Линейная аппроксимационная схема для многопроцессорной задачи Open Shop // Дискрет. анализ и исслед. операций. Сер. 1. – 1999. – Т.6. – № 2. – С. 3–22.

22. Генс Г. В., Левнер Е. В. Эффективные приближенные алгоритмы для комбинаторных задач: препринт. – М.: ЦЭМИ АН СССР, 1981. – 63 с.

23. Гимади Э. Х., Глебов Н. И., Перепелица В. А. Алгоритмы с оценками для задач дискретной оптимизации // Проблемы кибернетики. – М: Наука, 1975. – Вып. 31. – С. 35–42.

24. Гончаров Е. Н., Кочетов Ю. А. Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискрет. анализ и исслед. операций. Сер. 2. – 2002. – Т. 9. – № 2. – С. 13–30.

25. Гнеденко Б. В. Курс теории вероятностей. – М.: Наука, 1988. – 451 с.

26. Гришухин В. П. Оценка сложности алгоритма Балаша // Математические методы решения экономических задач. – М.: ЦЭМИ РАН, 1972. – Вып. 3. – С. 93–105.

27. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982. – 416 с.

28. Девятерикова М. В., Колоколов А. А., Колосов А. П. Унимодулярные преобразования для задач целочисленного программирования и анализ эффективности их применения // Тр. Ин-та математики и механики. – 2010. – Т. 16. – № 2. – С. 48–62.

29. Долгий А., Еремеев А.В. О сложности оптимальной рекомбинации для задачи об одномерной упаковке в контейнеры // Материалы VIII международной научно-технической конференции «Динамика систем, механизмов и машин». – Омск: Изд-во ОмГТУ, 2012. – Кн. 3. – С. 25–27.

30. Дюкова Е. В., Журавлёв Ю. И. Дискретный анализ признаковых описаний в задачах распознавания большой размерности //Ж. вычисл. матем. и матем. физ. – 2000. – Т. 40. – № 8. – С. 1264–1278.

31. Дюбин Г. Н., Корбут А. А. Поведение в среднем жадных алгоритмов для минимизационной задачи о ранце — общие распределения коэффициентов // Ж. вычисл. матем. и матем. физ. – 2008. – Т. 48. – № 9. – С. 1556–1570.

32. Евтушенко Ю. Г., Посыпкин М. А. Применение метода неравномерных покрытий для глобальной оптимизации частично целочисленных нелинейных задач // Ж. вычисл. матем. и матем. физ. – Т. 51. – № 8. – С. 1376–1389.

33. Емеличев В. А., Ковалев М. М., Кравцов М. К. Многогранники, графы, оптимизация (комбинаторная теория алгоритмов). – М.: Наука, 1982. – 344 с.

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

35. Еремеев А. В. О связи динамического программирования и многокритериальных эволюционных алгоритмов: препринт. – Омск: Изд-во ОмГУ, 2008. – 20 с.

36. Еремеев А. В. Эволюционные алгоритмы с возможностями динамического программирования // IV Всероссийская конференция «Проблемы оптимизации и экономические приложения»: материалы конференции. – Омск: Полигр. центр КАН, 2009. – С. 35–39.

37. Еремеев А. В. Вполне полиномиальная рандомизированная аппроксимационная схема на основе эволюционного алгоритма // Дискрет. анализ и исслед. операций. – 2010. – Т. 17. – № 4. – С. 3–17.

38. Еремеев А. В. О сложности оптимальной рекомбинации для задачи коммивояжера // Дискрет. анализ и исслед. операций. – 2011. – Т. 18. – № 1. – С. 27–40.

39. Еремеев А. В. Генетический алгоритм с турнирной селекцией как метод локального поиска // Дискрет. анализ и исслед. операций. – 2012. – Т. 19. – № 2. – С. 41–53.

40. Еремеев А. В., Заозерская Л. А., Колоколов А. А. Задача о покрытии множества: сложность, алгоритмы, экспериментальные исследования // Дискрет. анализ и исслед. операций. Сер. 2. – 2000. – Т. 7. – № 2. – С. 22–46.

41. Еремеев А. В., Ковалев М. Я., Кузнецов П. М. Приближенное решение задачи управления поставками со многими интервалами и вогнутыми функциями стоимости // Автомат. и телемех. – 2008. – № 7. – С. 90–97.

42. Еремеев А. В., Коваленко Ю. В. О задаче составления расписаний с группировкой машин по технологиям // Дискрет. анализ и исслед. операций. – 2011. – Т. 18. – № 5. – С. 54–79.

43. Еремеев А. В., Коваленко Ю. В. О сложности оптимальной рекомбинации для одной задачи составления расписаний с переналадками // Дискрет. анализ и исслед. операций. – 2012. – Т. 19. – № 3. – С. 13–26.

44. Еремеев А. В., Романова А. А., Сервах В. В., Чаухан С. С. Приближенное решение одной задачи управления поставками // Дискрет. анализ и исслед. операций. Сер. 2. – 2006. – Т. 13. – № 1. – С. 27–39.

45. Еремин И. И., Мазуров В. Д., Астафьев Н. Н. Несобственные задачи линейного и выпуклого программирования. – М.: Наука, 1983. – 336 с.

46. Журавлев Ю. И. Оценка сложности локальных алгоритмов для некоторых экстремальных задач на конечных множествах // Докл. АН СССР. – 1964. – Т. 158. – N 5. – С. 1018–1021.

47. Журавлев Ю. И. Теоретико-множественные методы алгебры логики // Проблемы кибернетики. – М.: Физматгиз, 1962. – Вып. 8. – С. 5–44.

48. Заблоцкая О. А. Нижняя оценка числа итераций для одного алгоритма лексикографической оптимизации // Дискретная оптимизация и численные методы решения прикладных задач. – Новосибирск: ВЦ СО АН СССР, 1986. – С. 21–27.

49. Забудский Г. Г., Лагздин А. Ю. Динамическое программирование для решения квадратичной задачи о назначениях на дереве // Автомат. и телемех. – 2012. – № 2. – С. 141-–155

50. Закревский А. Д. Логический синтез каскадных схем. – М.: Наука, 1981. – 414 с.

51. Заозерская Л. А. Об одном алгоритме перебора L-классов для решения задачи о покрытии множества // Труды XI Байкальской международной школы-семинара «Методы оптимизации и их приложения». – Иркутск: ИСЭМ СО РАН, 1998. – Т. 1. – С. 139–142.

52. Заозерская Л. А. Исследование и решение некоторых классов задач целочисленного программирования на основе регулярных разбиений: автореф. дис. канд. физ.-мат. наук. – Омск, 1998. – 16 с.

53. Заозерская Л. А., Колоколов А. А. Оценки среднего числа итераций для некоторых алгоритмов решения задачи об упаковке множества // Ж. вычисл. матем. и матем. физ. – 2010. – Т. 50. – № 2. – С. 242—248.

54. Зубков А. М., Попов Н. Н. Отношение частичного порядка, порожденное распределениями числа занятых ячеек // Матем. заметки. – 1982. – Т. 32. – № 1. – С. 97–102.

55. Ивахненко А. Г. Системы эвристической самоорганизации в технической кибернетике. – Киев: Техника, 1971. – 371 с.

56. Канторович Л. В., Залгаллер В. А. Рациональный раскрой промышленных материалов. – Новосибирск: Наука, 1971. – 299 с.

57. Кельманов А. В., Михайлова Л. В., Хамидуллин С. А. Об одной задаче поиска упорядоченных наборов фрагментов в числовой последовательности // Дискретн. анализ и исслед. операций, – 2009. – Т.16. – № 4. – С. 31–46.

58. Кемени Дж., Снелл Дж. Конечные цепи Маркова. – М.: Наука, 1970. – 272 с.

59. Китаев А., Шень А., Вялый М. Классические и квантовые вычисления. – М.: МЦНМО, ЧеРо, 1999. – 192 с.

60. Ковалев М. Я., Шафранский Я. М. Построение ε-приближенных алгоритмов оптимизации функции на последовательно конструируемых множествах // Журн. выч. матем. и матем. физ. – 1986. – Т. 26. – № 7. – С. 1006–1018.

61. Коган Д. И. Динамическое программирование и дискретная многокритериальная оптимизация: уч. пос. – Нижний Новгород: Изд-во Нижегородского ун-та, 2004. – 150 с.

62. Колоколов А. А. Регулярные разбиения в целочисленном программировании // Методы решения и анализа задач дискретной оптимизации. – Омск: Изд-во ОмГУ, 1992. – С. 67–93.

63. Колоколов А. А. Регулярные разбиения и отсечения в целочисленном программировании // Сиб. журн. исслед. операций. – 1994. – Т. 1. – № 2. – С. 18–39.

64. Колоколов А. А., Адельшин А.В., Ягофарова Д.И. Решение задачи выполнимости с использованием метода перебора L-классов // Информационные технологии. – 2009. – № 2. – С. 54–59.

65. Колоколов А. А., Косарев Н. А., Рубанова Н. А. Исследование отсечений Бендерса в декомпозиционных алгоритмах решения некоторых задач размещения // Омский научн. вестн. – 2005. – N 2. – С. 76–80.

66. Колоколов А. А., Леванова Т. В. Алгоритмы декомпозиции и перебора L-классов для решения некоторых задач размещения // Вестник Омского университета. – 1996. – № 1. – С. 21–23.

67. Колоколов А. А., Орловская Т. Г., Рыбалка М. Ф. Анализ алгоритмов целочисленного программирования с использованием L-разбиения и унимодулярных преобразований // Автомат. и телемех. – 2012. – № 2. – С. 178-–190.

68. Корбут А. А., Сигал И. Х., Финкельштейн Ю. Ю. Гибридные методы в дискретной оптимизации // Изв. АН СССР. Техн. кибернетика. – 1988. – № 1. – С. 65–77.

69. Корбут А. А., Финкельштейн Ю. Ю. Дискретное программирование. – М.: Наука, 1969. – 368 с.

70. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – М.: Вильямс, 2005. – 1296 с.

71. Кочетов Ю. А. Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения: сборник лекций молодежных научных школ по дискретной математике и ее приложениям. – М.: Изд-во центра прикладных исследований при механико-математическом факультете МГУ, 2001. – С. 84–117.

72. Кочетов Ю. А. Вычислительные возможности локального поиска в комбинаторной оптимизации // Журн. выч. матем. и матем. физ. – 2008. – Т. 48. – № 5. – С. 788–807.

73. 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.

74. Кочетов Ю. А., Плясунов А. В. Генетический локальный поиск для задачи о разбиении графа на доли ограниченной мощности // Журн. выч. матем. и матем. физ. – 2012. – Т. 52. – № 1. – С. 164–176.

75. Кузюрин Н. Н., Фомин С. А., Эффективные алгоритмы и сложность вычислений. – М.: МФТИ, 2007. – 326 с.

76. Лбов Г. С. Методы обработки разнотипных экспериментальных данных. – Новосибирск: Наука, 1981. – 160 с.

77. Леонтьев В. К. Верхняя оценка α-глубины (0,1)-матриц // Матем. заметки. – 1974. – Т. 15. – № 3. – С. 421–429.

78. Леонтьев В. К., Мамутов К. Х. Устойчивость решений в задачах линейного булева программирования // Ж. вычисл. матем. и матем. физ. – 1988. – Т. 28. – № 10. – С. 1475–1481.

79. Михалевич В. С., Шор Н. З. Численное решение многовариантных задач по методу последовательного анализа вариантов // Научно-методические материалы экономико-математического семинара «О численных методах решения многовариантных плановых и технико-экономических задач». – М.: Лаборатория по применению математических методов в экономических исследованиях и планировании АН СССР, 1962. – Вып. 1. – С. 15–41.

80. Моисеев Н. Н., Иванилов Ю. П., Столярова Е. М. Методы оптимизации. – М.: Наука, 1978. – 352 с.

81. Мухин В. И., Неймарк Ю. И., Ронин Е. И. Автоматная оптимизация с эволюционной адаптацией // Проблемы случайного поиска. – Рига: Зинатне, 1973. – С. 83–98.

82. Нечепуренко М. И., Попков В. К., Майнагашев С. М. и др. Алгоритмы и программы решения задач на графах и сетях. – Новосибирск: Наука, 1990. – 515 с.

83. Нигматуллин Р. Г. Метод наискорейшего спуска в задачах на покрытие // Вопросы точности и эффективности вычислительных алгоритмов: труды симпозиума. – Киев, 1969. – Вып. 5. – С. 116–126.

84. Норенков И. П. Эвристики и их комбинации в генетических методах дискретной оптимизации // Информационные технологии. – 1999. – № 1. – С. 2–7.

85. Попков В. К. Математические модели связности. – Новосибирск: ИВМ и МГ СО РАН, 2006. – 490 с.

86. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. – М.: Мир, 1984. – 512 с.

87. Растригин Л. А. Статистические методы поиска. – М.: Наука, 1968. – 376 с.

88. Растригин Л. А. Адаптация сложных систем. Методы и приложения. – Рига: Зинатне, 1981. – 375 с.

89. Редько В. Г., Цой Ю. Р. Оценка эффективности эволюционных алгоритмов // Доклады АН. – 2005. – Т. 404. – № 3. – С. 312–315.

90. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. – М.: Горячая линия-Телеком, 2006. – 452 с.

91. Сайко Л. А. Исследование мощности L-накрытий некоторых задач о покрытии // Дискретная оптимизация и анализ сложных систем: сб. научн. тр. – Новосибирск: ВЦ СО АН СССР, 1989. – С. 76–97.

92. Сапоженко А. А., Асратян А. С., Кузюрин Н. Н. Обзор некоторых результатов по задачам о покрытии // Методы дискретного анализа в решении комбинаторных задач. – Новосибирск: Ин-т математики СО АН СССР, 1977. – Вып. 30. – С. 46–75.

93. Сергиенко И. В. Математические модели и методы решения задач дискретной оптимизации. – Киев: Наук. думка, 1988. – 472 с.

94. Севастьянов Б. А. Курс теории вероятностей и математической статистики. – М.: Наука, 1982. – 256 с.

95. Семенкин Е. С. Эволюционные алгоритмы поддержки принятия решений при управлении сложными системами // Вестник Сибирского государственного аэрокосмического университета им. академика М.Ф. Решетнева. – 2005. – № 3. – С. 83–85.

96. Семенкина О. Э., Жидков В. В. Оптимизация управления сложными системами методом обобщенного локального поиска. – М.: МАКС Пресс, 2002. – 215 с.

97. Стрекаловский А.С. Элементы невыпуклой оптимизации – Новосибирск: Наука, 2003. – 356 с.

98. Схрейвер А. Теория линейного и целочисленного программирования: в 2 т. Т. 2: пер. с англ. – М.: Мир, 1991. – 342 с.

99. Тараканов В. Е. Комбинаторные задачи и (0,1)-матрицы. – М.: Наука, 1985. – 192 с.

100. Феллер В. Введение в теорию вероятностей и ее приложения. – М.: Мир, 1967. Т. 1. – 498 с.

101. Фогель Л., Оуэнс А., Уолш М. Искусственный интеллект и эволюционное моделирование. – М.: Мир, 1969. – 230 с.

102. Хачай М. Ю. Вычислительная сложность комбинаторных задач, индуцированных коллективными процедурами обучения распознаванию образов // Тр. ИММ УрО РАН. – 2010. – Т. 16. – № 3. – С. 276–284.

103. Шевченко В. Н. Качественные вопросы целочисленного программирования. – М.: Физматлит, 1995. – 190 с.

104. Щербина О. А. Методологические аспекты динамического программирования // Динамические системы. – 2007. – Вып. 22. – С. 21–36.

105. Эрдеш П., Спенсер Дж. Вероятностные методы в комбинаторике. – М.: Мир, 1976. – 137 с.

106. Ху Т. Целочисленное программирование и потоки в сетях. – М.: Мир, 1974. – 520 с.

107. Aarts E. H. L., Lenstra J. K. Introduction // Local Search in Combinatorial Optimization / Ed. by Aarts E. H. L., Lenstra J. K. – New York: John Wiley & Sons Ltd., 1997. – P. 1–19.

108. Aggarwal C. C., Orlin J. B., Tai R. P. An optimized crossover for maximum independent set // Operations Research. – 1997. – Vol. 45. – P. 225–234.

109. Aldous D., Vazirani U. U. "Go with the winners" algorithms // Proc. 35th Annual Symposium on Foundations of Computer Science (FOCS). – Los Alamitos, CA: IEEE Press, 1994. – P. 492–501.

110. Alekseeva E., Kochetov Yu., Plyasunov A. Complexity of local search for the p-median problem // Eur. J. Oper. Res. – 2008. – Vol. 191. – P. 736–752.

111. Alexandrov D., Kochetov Y. Behavior of the ant colony algorithm for the set covering problem // Proc. of Symp. on Oper. Res. (SOR'99) / Ed. by Inderfurth K. [et al.]. - Berlin: Springer, 2000. – P. 255–260.

112. Al-Sultan K., Hussain M., Nizami J. A Genetic algorithm for the set covering problem // J. Oper. Res. Soc. – 1996. – Vol. 47. – P. 702–709.

113. Antamoshkin A., Semenkin E. Local search efficiency when optimizing unimodal pseudoboolean functions // Informatica. – 1998. – Vol. 9. – N 3. – P. 279–296.

114. Arora S., Rabani U., Vazirani U. Simulating quadratic dynamical systems is PSPACE-complete // Proc. 26-th ACM Symp. on Theory of Computing. – Montreal: ACM, 1994. – P. 459–467.

115. Ausiello G., Crescenzi P., Gambosi G. et al. Complexity and

approximation: combinatorial optimization problems and their approximability properties. –Berlin: Springer-Verlag, 1999. – 529 p.

116. Ausiello G., Protasi M. Local search, reducibility and approximability of NP-optimization problems // Inform. Proc. Lett. – 1995. – Vol. 54. – P. 73–79.

117. Bäck T. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm // Proc. of Parallel Problem Solving from Nature / Ed. by Mánner R., Manderick B. – Amsterdam: North Holland, 1993. – P. 85–94.

118. Balas E. A sharp bound on the ratio between optimal integer and fractional covers // Math. Oper. Res. – 1984. – Vol. 9. – N 1. – P. 1 – 5.

119. Balas E., Carrera M. C. A dynamic subgradient-based branch and bound procedure for set covering // Oper. Res. – 1996. – Vol. 44. – N 6. – P. 875–890.

120. Balash E., Ho A. Set covering algorithms using cutting planes, heuristics and subgradient optimization: a computational study // Math. Programming Study. – 1980. – Vol. 12. – P. 37–60.

121. Balas E. and Niehaus W. Finding large cliques in arbitrary graphs by bipartite matching // DIMACS Series in Discrete Mathematics and Theoretical Computer Science / Ed. by Johnson D., Trick. M. – Providence, RI: American Mathematical Society, 1996.

122. Balas E., Niehaus W. Optimized crossover-based genetic algorithms for the maximum cardinality and maximum weight clique problems // J. Heuristics. – 1998. – Vol. 4. – N 2. – P. 107–122.

123. Barr R. S., Glover R. S., Klingman D. A new optimization method for

large scale fixed charge transportation problems // Operations Research. – 1981. – Vol. 29. – N 3. – P. 448–463.

124. Baybars I. A survey of exact algorithms for the simple assembly line balancing // Management Science. – 1986. – Vol. 32. – P. 909–932.

125. Beasley J. E. A Lagrangean heuristic for set covering problems // Naval Research Logist. – 1990. – Vol. 37. – P. 151–164.

126. Beasley J. E. OR-Library: distributing test problems by electronic mail // J. Oper. Res. Soc. – 1990. – Vol. 41. – N 11. – P. 1069–1072.

127. Beasley J. E., Chu P. C. A genetic algorithm for the set covering problem // Eur. J. Oper. Res. – 1996. – Vol. 94. – N 2. – P. 394–404.

128. Beasley J. E., Jörnsten K. Enhancing an algorithm for set covering problems // Eur. J. Oper. Res. – 1992. – Vol. 58. – P. 293–300.

129. Becker C., Scholl A. A survey on problems and methods in generalized assembly line balancing // Eur. J. Oper. Res. – 2006. – Vol. 168. – N 31. – P. 694–715.

130. Bellman R. E, Dreyfus S. E. Applied dynamic programming. – Princeton, N.J: Princeton University Press, 1962. – 363 p.

131. Bertele U., Brioschi F. Nonserial Dynamic Programming. – New York: Academic Press, 1972. – 235 p.

132. Beyer H.-G., Schwefel H.-P., Wegener I. How to analyse evolutionary algorithms // Theor. Comp. Sci. – 2002. – Vol. 287. – P. 101–130.

133. Borisovsky P., Dolgui A., Eremeev A. Genetic algorithms for supply management problem with lower-bounded demands // Inform. Control Problems in Manufact.: Proc. of 12th IFAC Intern. Symp. / Ed. by Dolgui A. [et al.]. – St Etienne: Elsevier, 2006. – Vol. 3. – P. 521–526.

134. Borisovsky P., Dolgui A., Eremeev A. Genetic algorithms for a supply management problem: MIP-recombination vs greedy decoder // Eur. J. Oper. Res. – 2009. – Vol. 195. – N 3. – P. 770–779.

135. Borisovsky P. A., Eremeev A. V. On performance estimates for two evolutionary algorithms // Applications of Evolutionary Computing: Proc. of EvoWorkshops 2001 / Ed. by Boers E. J. W. [et al.]. – Berlin: Springer-Verlag, 2001. – P. 161–171. – (Lect. Notes Comput. Sci.; Vol. 2037).

136. Borisovsky P. A., Eremeev A. V. A study on performance of the (1+1)-evolutionary algorithm // Foundations of Genetic Algorithms 7 / Ed. by De Jong K., Poli R., Rowe J. – San Francisco: Morgan Kaufmann, 2003. – P. 271–287.

137. Borisovsky P. A., Eremeev A. V. Comparing evolutionary algorithms to the (1+1)-EA // Theor. Comp. Sci. – 2008. – Vol. 403. – N 1. – P. 33–41.

138. Borisovsky P. A., Eremeev A. V., Floudas C. A. et al. A hybrid method for multi-product continuous plant scheduling based on decomposition approach and genetic algorithm // Abstracts of GOR Workshop «Scheduling in the Process Industry». Bad Honnef, 2008. – P. 5.

139. Borisovsky P. A., Zavolovskaya M. S. Experimental comparison of two evolutionary algorithms for the independent set problem // Applications of Evolutionary Computing: Proc. of EvoWorkshops 2003 / Ed. by Cagnoni S. [et al.]. – Berlin: Springer-Verlag, 2003. – P. 154–164. – (Lect. Notes Comput. Sci.; Vol. 2611).

140. Bovet D.P., Crescenzi P. Introduction to the theory of complexity. – New York: Prentice-Hall, 1993. – 290 p.

141. Caprara A., Fischetti M., Toth P. Heuristic method for the set covering problem // Operations Research. – 1999. – Vol. 47. – N 5. – P. 730–743.

142. Chauhan S. S., Eremeev A. V., Kolokolov A. A., Servakh V. V. Concave cost supply management for single manufacturing unit // Supply chain optimisation. Product/process design, factory location and flow control. – N. Y.: Springer-Verlag, 2005. – P. 167–174. – (Applied Optim.; Vol. 94).

143. Chauhan S. S., Eremeev A. V., Romanova A.A., Servakh V. V. Approximation of linear cost supply management problem with lower-bounded demands // Proc. of Discrete Optimization Methods in Production and Logistics (DOM'2004). – Omsk: Nasledie Dialog-Sibir, 2004. – P. 16–21.

144. Chauhan S. S., Eremeev A. V., Romanova A. A., Servakh V. V., Woeginger G. J. Approximation of the supply scheduling problem // Oper. Res. Lett. – 2005. – Vol. 33. – N 3. – P. 249–254.

145. Chauhan S. S., Proth J.-M. The concave cost supply problem // Eur. J. Oper. Res. – 2003. – Vol. 148. – N 2. – P. 374–383.

146. Chvátal. V. A Greedy Heuristic for the Set Covering Problem // Mathematics of Oper. Res. – 1979. – Vol. 4. – N 3. – P. 233–235.

147. Cook W., Seymour P. Tour merging via branch-decomposition // INFORMS J. Comput. – 2003.– Vol. 15. – N 2. – P. 233–248.

148. Cotta C. A study on allelic recombination // Proc. of 2003 Congress on Evolutionary Computation. – Canberra: IEEE Press, 2003. – P. 1406–1413.

149. Cotta C., Alba E., Troya J. M. Utilizing dynastically optimal forma recombination in hybrid genetic algorithms // Proc. of the 5th Int. Conf. on Parallel Problem Solving from Nature. – Berlin: Springer-Verlag, 1998. – P. 305–314. – (Lect. Notes Comput. Sci.; Vol. 1498).

150. Craig C. C. Use of marked specimens in estimating populations // Biometrika. – 1953. – Vol. 40. – N 1–2. – P. 170–176.

151. Crainic T. G., Gendreau M. Cooperative parallel tabu search for capacitated network design // J. Heuristics. – 2002. – Vol. 8. – P. 601–627.

152. Daley D. J. Stochastically monotone Markov chains // Z. Wahrscheinlickeitstheorie und Verw. Gebiete. – 1968. – Vol. 10. – P. 307–317.

153. Danna E., Rothberd E., Le Pape C. Exploring relaxation induced neighborhoods to improve MIP solutions // Mathematical Programming. Ser. A. – 2005. – Vol. 102. – P. 71–90.

154. Darroch J. N. The multiple-recapture census. I: estimation of a closed population // Biometrika. – 1958. – Vol. 45. – N 3–4. – P. 343–359.

155. Doerr B., Eremeev A., Horoba C., Neumann F., Theile M. Evolutionary algorithms and dynamic programming // Proc. of Genetic and Evolutionary Computation Conf. (GECCO). – New York: ACM Press, 2009. – P. 771–777.

156. Doerr B., Eremeev A., Neumann F., Theile M., Thyssen C. Evolutionary algorithms and dynamic programming // Theor. Comp. Sci. – 2011. – Vol. 412. – P. 6020–6035.

157. Doerr B., Happ E., Klein C. Crossover can provably be useful in evolutionary computation // Theor. Comp. Sci. – 2012. – Vol. 425. – P. 17-33.

158. Doerr B., Hebbinghaus N., Neumann F. Speeding up evolutionary algorithms through asymmetric mutation operators // Evolutionary Computation. – 2007. – Vol. 15. – N 4. – P. 401–410.

159. Doerr B., Klein C., Storch T. Faster evolutionary algorithms by superior graph representations // Proc. IEEE Symp. on Foundations of Comput. Intelligence (FOCI). – Los Alamitos, CA: IEEE Press, 2007. – P. 245–250.

160. Dolgui A., Eremeev A., Guschinskaya O. MIP-based GRASP and genetic algorithm for balancing transfer lines // Matheuristics. Hybridizing Metaheuristics and Mathematical Programming / Ed. by Maniezzo V., Stutzle T., Voss S. – Berlin: Springer-Verlag, 2010. – P. 189–208.

161. Dolgui A., Eremeev A., Kolokolov A., Sigaev V. A Genetic Algorithm for the Allocation of Buffer Storage Capacities in a Production Line with Unreliable Machines // Journal of Mathematical Modelling and Algorithms. – 2002. – Vol. 1. – N 2. – P. 89–104.

162. Dolgui A., Eremeev A. V., Kovalyov M. Y., Kuznetsov P. M. Multi-product lot sizing and scheduling on unrelated parallel machines // IIE Transactions. – 2010. – Vol. 42. – N 7. – P. 514–524.

163. Dolgui A., Eremeev A. V., Sigaev V. S. HBBA: hybrid algorithm for buffer allocation in tandem production lines // Journal of Intelligent Manufacturing. – 2007. – Vol. 18. – N 3. – P. 411–420.

164. Dolgui A., Finel B., Guschinsky N., Levin G., Vernadat F. A heuristic approach for transfer lines balancing // Journal of Intelligent Manufacturing. – 2005. – Vol. 16. – N 2. – P. 159–171.

165. Dolgui A., Finel B., Guschinsky N., Levin G., Vernadat F. MIP approach to balancing transfer lines with blocks of parallel operations // IIE Transactions. – 2006. – Vol. 38. – N 10. – P. 869–882.

166. Dolgui A., Guschinsky N., Levin G. Approaches to balancing of transfer lines with blocks of parallel operations: Preprint No 8; – Minsk: Institute of Engineering Cybernetics of Academy of Sciences of Belarus, 2000. – 42 p.

167. Dolgui A., Guschinsky N., Levin G. A special case of transfer lines balancing by graph approach // Eur. J. Oper. Res. – 2006. – Vol 168. – N 3. – P. 732–746.

168. Dongarra J. J. Performance of various computers using standard linear equations software: report No. CS-89-85. – Tennessee: University of Tennessee, 2003. – 78 p.

169. Droste S., Jansen T., Wegener I. Upper and lower bounds for randomized search heuristics in black-box optimization // Theory of Computing Systems. – 2006. – Vol. 39. – N 4. – P. 525–544.

170. Eckert C., Gottlieb J. Direct representation and variation operators for the fixed charge transportation problem // Proc. of the 7th International Conference on Parallel Problem Solving from Nature. – Berlin: Springer-Verlag, 2002. – P. 77–87. – (Lect. Notes Comput. Sci.; Vol. 2439).

171. Eppstein D. The traveling salesman problem for cubic graphs // Journal of Graph Algorithms and Applications. – 2007. – Vol. 11. – N 1. – P. 61–81.

172. Erdös P. On a combinatorial problem I // Nordisk Mat. Tidskrift. – 1963. – Vol. 11. – P. 5–10.

173. Eremeev A. V. A Genetic algorithm with a non-binary representation for the set covering problem // Proc. of Operations Research (OR'98). – Berlin: Springer-Verlag, 1999. – P. 175–181.

174. Eremeev A. V. Modeling and analysis of genetic algorithm with tournament selection // Proc. of Artificial Evolution Conference (AE'99) / Ed. by Fonlupt C. [et al.] – Berlin: Springer Verlag, 2000. – P. 84–95. – (Lect. Notes Comput. Sci.; Vol. 1829).

175. Eremeev A. V. On complexity of optimal recombination for binary representations of solutions // Evolutionary Computation. – 2008. – Vol. 16. – N 1. – P. 127–147.

176. Eremeev A. V. On complexity of optimal recombination for the travelling salesman problem // Proc. of Evolutionary Comput. in Combinatorial

Optimization (EvoCOP 2011) / Ed. by P. Merz J.-K. Hao. – Berlin: Springer Verlag, 2011. – P. 215–225. – (Lect. Notes Comput. Sci.; Vol. 6622).

177. Eremeev A. V., Kolokolov A. A. On some genetic and L-class enumeration algorithms in integer programming // Proc. of the First International Conference on Evolutionary Computation and its Applications. – Moscow: Russian Academy of Sciences, 1996. – P. 297–303.

178. Eremeev A. V., Kolokolov A. A., Zaozerskaya L. A. A hybrid algorithm for set covering problem // Proc. of International Workshop on Discrete Optimization Methods in Scheduling and Computer-Aided Design. – Minsk: National Academy of Sciences Belarus, 2000. – P. 123–129.

179. Eremeev A. V., Reeves C. R. Evolutionary algorithms in discrete optimisation // Book of abstracts of Discrete Analysis and Oper. Res. Conf. – Novosibirsk: Institute of Mathematics, 2002. – P. 40–45.

180. Eremeev A. V., Reeves C. R. Non-parametric estimation of properties of combinatorial landscapes // Applications of Evolutionary Computing: Proc. of EvoWorkshops 2002 / Ed. by Cagnoni S., Gottlieb J., Hart E., Middendorf M., Raidl G. – Berlin; Heidelberg: Springer-Verlag, 2002. – P. 31–40. – (Lect. Notes Comput. Sci.; Vol. 2279).

181. Eremeev A. V., Reeves C. R. On confidence intervals for the number of local optima // Applications of Evolutionary Computing: Proc. of EvoWorkshop 2003 / Ed. by Raidl G. R., Meyer J.-A., Middendorf M., Cagnoni S., Cardalda J. J. R., Corne D., Gottlieb J., Guillot A., Hart E., Johnson C.G., Marchiori E. – Heidelberg: Springer-Verlag, 2003. – P. 224–235. – (Lect. Notes Comput. Sci.; Vol. 2611).

182. Eremeev A. V. Non-elitist genetic algorithm as a local search

method: Preprint (arXiv:1307.3463v2 [cs.NE]). – Cornell: Cornell University, 2013. – 9 p. URL: http://arxiv.org/abs/1307.3463.

183. Feo T. A., Resende M. G. C. Greedy randomized adaptive search procedures // J. of Global Optimization. – 1995. – Vol. 6. – P. 109–133.

184. Festa P., Resende M. G. C. GRASP: An annotated bibliography // Essays and surveys on metaheuristics / Ed. by Ribeiro C. C., Hansen P. – Norwell, MA: Kluwer Acad. Pbs., 2001. – P. 325–367.

185. Fisher M. L., Kedia P. Optimal solution of set covering problem using dual heuristics // Manag. Sci. – 1990. – Vol. 36. – N 8. – P. 674–688.

186. Friedrich T., Hebbinghaus N., Neumann F., He J., Witt C. Approximating covering problems by randomized search heuristics using multi-objective models // Proc. of Genetic and Evolutionary Computation Conf. (GECCO). – New York: ACM Press, 2007. – P. 797–804.

187. Fulkerson D. R., Nemhauser G. L., Trotter L. E. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems // Math. Prog. Study. – 1974. – Vol. 2. – P. 72–81.

188. Galil Z., Park K. Parallel algorithms for dynamic programming recurrences with more than $O(1)$ dependency // J. Parallel Distrib. Comput. – 1994. – Vol. 21. – N 2. – P. 213–222.

189. Garnier J., Kallel L. How to detect all maxima of a function? // Proc. of the 2nd EVONET Summer School on Theor. Aspects of Evolutionary Comput. (Anvers, 1999). – Berlin: Springer-Verlag, 2001. – P. 343–370.

190. Gaspero L. D., Gärtner J., Kortsarz G., Musliu N., Schaerf A., Slany W. The minimum shift design problem // Annals of Oper. Res. – 2007. – Vol. 155. – N 1. – P. 79–105.

191. Garthwait P. H., Buckland S. T. Analysis of multiple recapture census by computing conditional probabilities // Biometrics. – 1990. – Vol. 46. – N 1. – P. 231–238.

192. Ghosh S., Gagnon R. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly lines // International Journal of Production Research. – 1989. – Vol. 27. – N 4. – P. 637–670.

193. Giel O., Wegener I. Evolutionary algorithms and the maximum matching problem // Proc. of the 20th Ann. Symp. on Theoretical Aspects of Computer Science (STACS 2003). – Berlin: Springer-Verlag, 2003. – P. 415–426. – (Lect. Notes Comput. Sci.; Vol. 2607).

194. Glover F., Laguna M. Tabu Search. – Norwell, MA: Kluwer Acad. Pbs., 1997. – 408 p.

195. Glover F., Laguna M., Marti R. Fundamentals of scatter search and path relinking // Control and Cybernetics. – 2000. – Vol. 29. – N 3. – P. 653–684.

196. Glover F., Sherali H. Some classes of valid inequalities and convex hull characterizations for dynamic fixed-charge problems under nested constraints // Annals of Oper. Res. – 2005. – Vol 140. – N 1. – P. 215–233.

197. Goldberg D. E. Genetic Algorithms in search, optimization and machine learning. – Reading, MA: Addison-Wesley, 1989. – 412 p.

198. Golay M. J. E. Series for low-autocorrelation binary sequences // IEEE Trans. Inform. Theory. – 1977. – Vol. 23. – P. 43–51.

199. Goldberg D. E. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing // Complex Systems. – 1990. – Vol. 4. – P. 445–460.

200. Goldberg D. E., Lingle R. Alleles, loci and the traveling salesman problem // Proc. of International Conference on Genetic Algorithms and Their Applications // Ed. by Grefenstette J. J. – Hillsdale, NJ: Lawrence Erlbaum Associates, 1985. – P. 154–159.

201. Goldberg M. K., Russell H. Toward Computing m(4) // Ars Combinatoria. – 1995. – Vol. 3. – N 9. – P. 139–148.

202. Goossens D., Maas A., Spieksma F., van de Klundert J. Exact algorithms for procurement problems under a total quantity discount structure // Eur. J. Oper. Res. – 2007. – Vol. 178. – N 2. – P. 603–626.

203. Gottlieb J., Paulmann L. Genetic algorithms for the fixed charge transportation problem // Proc. of the 5th IEEE Intern. Conf. on Evolutionary Comput. – Anchorage, AK: IEEE Press, 1998. – P. 330–335.

204. Croes G. A. A method for solving Traveling-Salesman problems // Operations Research. – 1958. – Vol. 6. – N 6. – P. 791–812.

205. Grossman T., Wool A. Computational experience with approximation algorithms for the set covering problem // Eur. J. Oper. Res. – 1997. – Vol. 101. – N 1. – P. 81–92.

206. Guruswami V., Hastad J., Sudan M. Hardness of approximate hypergraph coloring // SIAM J. Comput. – 2002. – Vol. 31. – N 6. – P. 1663–1686.

207. Guschinskaya O., Dolgui A. A comparative evaluation of exact and heuristic methods for transfer lines balancing problem // Information Control Problems in Manufacturing 2006: A Proceedings volume from the 12th IFAC International Symposium / Ed. by Dolgui A., Morel G. and Pereira C. – St Etienne, France: Elsevier Science, 2006. – Vol. 2. – P. 17–19.

208. Guschinskaya O., Dolgui A., Guschinsky N., Levin G. A heuristic multi-

start decomposition approach for optimal design of serial machining lines // Eur. J. Oper. Res. – 2008. – Vol. 189. – N 3. – P. 902–913.

209. Guschinskaya O., Gurevsky E., Dolgui A., Eremeev A. Metaheuristic approaches for the design of machining lines // International Journal of Advanced Manufacturing Technology. – 2011. – Vol. 55. – N 1. – P. 11–22.

210. Hart J. P., Shogan A. W. Semi-greedy heuristics: An empirical study // Oper. Res. Lett. – 1987. – Vol. 6. – P. 107–114.

211. Held M., Karp R. M. A dynamic programming approach to sequencing problems // J. of Soc. for Indust. and Appl. Math. – 1962. – Vol. 10. – P. 196–210.

212. Hochbaum D. S. Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems // Approximation Algorithms for NP-Hard Problems / Ed. by Hochbaum D. – Boston: PWS Publishing Company, 1997. – P. 94–143.

213. Holland J. Adaptation in natural and artificial systems. – Ann Arbor: University of Michigan Press, 1975. – 183 p.

214. Horoba C. Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem // Proc. of 10th International Workshop on Foundations of Genetic Algorithms (FOGA). – New York: ACM Press, 2009. – P. 113–120.

215. Ibarra O. H., Kim C. E. Fast approximation algorithms for the knapsack and sum of subset problems // J. ACM. – 1975. – Vol. 22. – N 4. – P. 463–468.

216. Il'ev V. P. Hereditary systems and greedy-type algorithms // Discrete Applied Mathematics – 2003. – Vol. 132. – N 1–3. – P. 137–148.

217. Itai A., Papadimitriou C. H., Szwarcfiter J. L. Hamilton paths in grid graphs // SIAM J. Comput. – 1982. – Vol. 11. – N 4. – P. 676–686.

218. Janak S. L., Floudas C. A., Kallrath J., Vormbrock N. Production scheduling of a large-scale industrial batch plant. I. short-term and medium- term scheduling // Ind. Eng. Chem. Res. – 2006. – Vol. 45. – P. 8234–8252.

219. Jansen T., Wegener I. On the utility of populations in evolutionary algorithms // Proc. of Genetic and Evolutionary Computation Conf. (GECCO). – New York: ACM Press, 2001. – P. 1034–1041.

220. Jansen T., Wegener I. On the analysis of evolutionary algorithms – a proof that crossover really can help // Algorithmica. – 2002. – Vol. 34. – N 1. – P. 47–66.

221. Jeroslow R. G. Cutting-plane theory: algebraic methods // Discrete Math. – 1978. – Vol. 23. – N 2. – P. 121–151.

222. Jerrum M., Sinclair A. Polynomial-time approximation algorithms for the Ising model // SIAM J. Comput. – 1993. – Vol. 22. – N 5. – P. 1087–1116.

223. Jerrum M., Sinclair A., Vigoda E. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries // J. of the ACM. – 2004. – Vol. 51. – N 4. – P. 671-–697.

224. Johnson D. S., Trick M. A.: Introduction to the second DIMACS challenge: cliques, coloring, and satisfiability // DIMACS Series in Discrete Math. and Theoretical Comput. Sci. / Ed. by Johnson D., Trick. M. – Providence, RI: American Math. Soc., 1996. – Vol. 26. – P. 1–10.

225. Kauffman S. A. Adaptation on rugged fitness landscapes // Lectures in the Sciences of Complexity. SFI Studies. – Redwood City: Addison-Wesley, 1989. – Vol. 1. – P. 619–712.

226. Khachay M. Yu. On the computational complexity of the minimum committee problem // J. Math. Mod. and Alg. – 2007. – Vol. 6. – N 4. – P. 547–561.

227. Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by simulated annealing // Science. – 1983. – Vol. 220. – P. 671–680.

228. Klötzler R. Multiobjective dynamic programming // Mathematische Operationsforschung und Statistik. Series Optimization. – 1978. – Vol. 9. – P. 423–426.

229. Ko K. Some observations on the probabilistic algorithms and NP-hard problems // Information Processing Letters. – 1982. – Vol. 14. – P. 39–43.

230. Korte B., Vygen J. Combinatorial Optimization. Theory and Algorithms. – Third Ed. – Berlin: Springer-Verlag, 2005. – 588 p.

231. Koza J. R. Genetic Programming II: Automatic Discovery of Reusable Programs (Complex Adaptive Systems). – Cambridge, MA: MIT Press, 1994. – 768 p.

232. Krarup J., Pruzan P. The simple plant location problem: survey and synthesis // Eur. J. Oper. Res. – 1983. – Vol. 12. – P. 36–81.

233. Kubiak W., Cheng J., Kovalyov M. Y. Fast fully polynomial approximation schemes for minimizing completion time variance // Eur. J. Oper. Res. – 2002. – Vol. 137. – N 2. – P. 303–309.

234. Laumanns M., Thiele L., Zitzler E. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions // IEEE Transact. on Evol. Comput. – 2004. – Vol. 8. – Issue 2. – P. 170–182.

235. Lenstra, H. W. Jr. Integer programming with a fixed number of variables // Math. of Oper. Res., – 1983. – Vol. 8. – N 4. – P. 538–548.

236. Lin S. Computer solutions of the Traveling Salesman Problem // Bell System Tech. J. – 1965. – Vol. 44. – N 10. – P. 2245–2269.

237. Lourenço H., Paixão J., Portugal R. Multiobjective metaheuristics for the bus driver scheduling problem // Transportation Sci. – 2001. – Vol. 35. –P. 331–343.

238. Mao C. X. On the nonidentifiability of population sizes // Biometrics. – 2008. – Vol. 64. – P. 977–981.

239. Mannino C., Sassano A. Solving hard set covering problems // Oper. Res. Lett. – 1995. – Vol. 18. – P. 1–5.

240. Michalewicz Z. Genetic algorithms + data structures = evolution programs. – Berlin; Heidelberg; New York: Springer-Verlag, 1996. – 387 p.

241. Mitten L. G. Composition principles for synthesis of optimal multistage processes // Operations Research. – 1964. – Vol. 12. – N 4. – P. 610–619.

242. Mood A. M., Graybill F. A., Boes D. C. Introduction to the theory of statistics. – Third Ed. – New York: McGraw-Hill, 1973. – 577 p.

243. Moscato P. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms: technical report Caltech concurrent computation program: report 826. – Pasadena, CA: Caltech, 1989. – 67 p.

244. Mühlenbein H. How genetic algorithms really work: I. mutation and hillclimbing // Proc. of Parallel Problem Solving from Nature / Ed. by Mánner R., Manderick B. – Amsterdam: North Holland, 1993. – P. 15–26.

245. Neumann F. Expected runtimes of evolutionary algorithms for the Eulerian cycle problem // Comput. Oper. Res. – 2008. – Vol. 35. – N 9. – P. 2750–2759.

246. Neumann F., Reichel J. Approximating minimum multicuts by evolutionary multi-objective algorithms // Proc. of the 10th Int. Conf. on Parallel Problem Solving from Nature. – Berlin: Springer-Verlag, 2008. – P. 72–81. – (Lect. Notes Comput. Sci.; Vol. 5199).

247. Neumann F., Reichel J., and Skutella M. Computing minimum cuts by randomized search heuristics // Algorithmica. – 2011. – Vol. 59. – P. 323–342.

248. Neumann F., Witt C. Bioinspired computation in combinatorial optimization – algorithms and their computational complexity. – Berlin: Springer-Verlag, 2010. – 216 p.

249. Neumann F., Wegener I. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem // Theor. Comp. Sci. – 2007. – Vol. 378. – N 1. – P. 32–40.

250. Ng C. T., Kovalyov M. Y., Cheng T. C. E. An FPTAS for a supply scheduling problem with non-monotone cost functions // Naval Res. Logistics. – 2008. – Vol. 55. – P. 194–199.

251. Nix A., Vose M. D. Modeling genetic algorithms with Markov chains // Annals of Math. and Artificial Intelligence. – 1992. – Vol. 5. – P. 79–88.

252. Nourie F. J., Venta E. R. An upper bound on the number of cuts needed in Gomory's method of integer forms // Oper. Res. Lett. – 1982. – Vol. 1. – N. 4. – P.129–133.

253. Odijk M. A., van Maaren H. Improved solutions to the Steiner triple covering problem // Inform. Processing Lett. – 1998. – Vol. 65. – P.67–69.

254. Ohlsson M., Peterson C., Söderberg B. An efficient mean field approach to the set covering problem // Eur. J. Oper. Res. – 2001. – Vol. 133. – P. 583–595.

255. Ortega F., Wolsey L. A branch-and-cut algorithm for the single commodity uncapacitated fixed-charge network flow problem // Networks. – 2003. – Vol. 41. – N 3. – P. 143–158.

256. Ostrowski J., Linderoth J. T., Rossi F., Smriglio S. Solving steiner triple covering problems // Oper. Res. Lett. – 2011. – Vol. 39. – P. 127–131.

257. Peschel M., Riedel C. Use of vector optimization in multiobjective decision making // Conflicting Objectives in Decisions / Ed. by Bell D. E., Keeney R. L., and Raiffa H. – Chichester: Wiley, 1977. – P. 97-–121.

258. Pickands J., Raghavachari M. Exact and asymptotic inference for the size of a population // Biometrika. – 1987. – Vol. 74. – N 2. – P. 355–363.

259. Pledger S. The performance of mixture models // Heterogeneous Closed Population Capture-Recapture. – 2005. – Vol. 61. – N 3. – P. 868–876.

260. Rabani Y., Rabinovich Y., Sinclair A. A computational view of population genetics // Random Struct. and Algor. – 1998. – Vol. 12. – P. 314–334.

261. Radcliffe N. J. The algebra of genetic algorithms // Annals of Math. and Artificial Intelligence. – 1994. – Vol. 10. – N 4. – P. 339-384.

262. Raidl G. R., Julstrom B. A. Edge sets: An effective evolutionary coding of spanning trees // IEEE Transactions on Evolutionary Comput. – 2003. – Vol. 7. – N 3. – P. 225–239.

263. Sahoo S., Albrecht A. A. Approximating the set of local minima in partial RNA folding landscapes // Bioinformatics. – 2012. – Vol. 28. – Issue 4. – P. 523–530.

264. Seber G. A. F. The estimation of animal abundance. – London: Charles Griffin, 1973. – 506 p.

265. Rastrigin L.A. Random search in evolutionary computations // Proc. First International Conference on Evolutionary Computation and its

Applications. – Moscow: Russian Academy of Sciences, 1996. – P. 135–142.

266. Rechenberg I. Evolutionsstrategie: optimerung technischer systeme nach prinzipen der biologischen evolution. – Stuttgart: Formann-Holzboog Verlag, 1973. – 170 p.

267. Reichel J., Skutella M. Evolutionary algorithms and matroid optimization problems // Proc. Genetic and Evolutionary Comput. Conf. (GECCO). – New York: ACM Press, 2007. – P. 947–954.

268. Reeves C. R. The "crossover landscape" and the hamming landscape for binary search spaces // Foundations of Genetic Algorithms 7 / Ed. by De Jong K., Poli R., Rowe J. – San Francisco: Morgan Kaufmann, 2003. – P. 81–98.

269. Reeves C. R. Genetic algorithms for the operations researcher // INFORMS J. Comput. – 1997. – Vol. 9. – N 3. – P. 231–250.

270. Reeves C. R. Estimating the number of optima in a landscape. Part I: statistical principles: technical report SOR#01-03 / Coventry University. – Conventry: Coventry University, 2001. – 19 p.

271. Reeves C. R.: Estimating the number of optima in a landscape. Part II: experimental investigations: technical report SOR#01-04 / Coventry University. – Conventry: Coventry University, 2001. – 18 p.

272. Reeves C. R., Eremeev A. V. Statistical analysis of local search landscapes // J. Oper. Res. Soc. – 2004. – Vol. 55. – N 7. – P. 687–693.

273. Reeves C. R., Rowe J. E. Genetic algorithms: principles and perspectives. – Norwell, MA: Kluwer Acad. Pbs., 2002. – 333 p.

274. Resende M.G.C., Ribeiro C. C. Greedy randomized adaptive search

procedures. // Handbook of metaheuristics / Ed. by Glover F., Kochenberger G. – Norwell, MA: Kluwer Acad. Pbs., 2003. – P. 219–249.

275. Rödl V., Tovey C. Multiple optima in local search // J. Algorithms. – 1987. – Vol. 8. – P. 250–259.

276. Rudolph G. Convergence analysis of canonical genetic algorithms // IEEE Transactions on Neural Networks. – 1994. – Vol. 5. – N 1. – P. 96–101.

277. Rudolph G. Finite Markov chain results in evolutionary computation: A tour d'horizon // Fundamenta Informaticae. – 1998. – Vol. 35. – N 1–4. – P. 67–89.

278. Rudolph G. Evolutionary search for minimal elements in partially ordered finite sets // Proc. of the 7th Annual Conference on Evolutionary Programming / Ed. by Porto V. W., Saravanan N., Waagen D., and Eiben A.E. – Berlin: Springer, 1998. – P. 345—353.

279. Scharnow J., Tinnefeld K., Wegener I. The analysis of evolutionary algorithms on sorting and shortest paths problems // J. Math. Mod. and Alg. – 2004. – Vol. 3. – P. 349–366.

280. Schnabel Z. E. The estimation of the total fish population of a lake // American Math. Monthly. – 1938. – Vol. 45. – P. 348–352.

281. Scholl A. Balancing and sequencing of assembly lines. – Heidelberg: Physica-Verlag, 1999. – 318 p. (Contributions to Management Sci.; Vol. 29).

282. Scholl A., Klein R. Balancing assembly lines effectively – A computational comparison // Eur. J. Oper. Res. – 1999. – Vol. 114. – N 1. – P. 50–58.

283. Schuurman P., Woeginger G. Approximation schemes – a tutorial [Электронный ресурс] / Eindhoven University of Technology, 2006. – 68 p. –

Режим доступа: http://www.win.tue.nl/∼gwoegi/papers (дата обращения: 21.08.2012).

284. Sheng S., Dechen Z., Xiaofei X. Genetic algorithm for the transportation problem with discontinuous piecewise linear cost function // Int. J. of Comput. Sci. and Network Secur. – 2006. – Vol. 6. – N 7a. – P. 181–190.

285. Sun M., Aronson J., McKeown P. Drinka D., A tabu search heuristic procedure for the fixed charge transportation problem // Eur. J. Oper. Res. – 1998. – Vol. 106. – P. 441–456.

286. Theile M. Exact solutions to the travelling salesperson problem by a population-based evolutionary algorithm // Proc. Europ. Conf. on Evolutionary Comp. in Combinatorial Optimisation (EvoCOP). – Berlin: Springer-Verlag, 2009. – P. 145–155. – (Lect. Notes Comp. Sci.; Vol. 5482).

287. Vercellis C. A probabilistic analysis of the set covering problem // Annals of Oper. Res. – 1984. – Vol. 1. – P.255–271.

288. Vitányi P. M. B. A discipline of evolutionary programming // Theor. Comp. Sci. – 2000. – Vol. 241. – N 1–2. – P. 3–23.

289. Vose M. D. Modeling simple genetic algorithms // Evolutionary Comput. – 1995. – Vol. 3. – N 4. – P. 453–472.

290. Vose M. D., Wright A. H. The walsh transform and the theory of the simple genetic algorithm // Genetic algorithms for pattern recognition / Ed. by Pal S., Wang P. – Boca Raton, FL: CRC Press, 1995. – P. 25–44.

291. Vose M. D., Wright A. H. Stability of vertex fixed points and applications // Foundations of Genetic Algorithms 3 / Ed. by Whitley D., Vose M. – San Mateo, CA: Morgan Kaufmann, 1995. – P. 103–114.

292. Whitley D., Hains D. and Howe A. A hybrid genetic algorithm for the traveling salesman problem using generalized partition crossover // Proc.

of the 11th Int. Conf. on Parallel Problem Solving from Nature. – Berlin: Springer-Verlag, 2010. P. 566–575. – (Lect. Notes Comput. Sci.; Vol. 6238)

293. Woeginger G. J. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? // INFORMS J. Comput. – 2000. – Vol. 12. – N 1. – P. 57–74.

294. Yagiura M., Ibaraki T. The use of dynamic programming in genetic algorithms for permutation problems // Eur. J. Oper. Res. – 1996. – Vol. 92. – P. 387–401.

295. Yagiura M., Kishida M., Ibaraki. T. A 3-flip neighborhood local search for the set covering problem // Eur. J. Oper. Res. – 2006. – Vol. 172. – P. 472–499.

296. Yannakakis M. Computational complexity // Local Search in Combinatorial Optimization / Ed. by Aarts E.H.L., Lenstra J.K. – Chichester: Wiley, 1997. – P. 19–55.