МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи
УДК 519.21

РУДНЕВ Антон Сергеевич

Алгоритмы локального поиска для задач двумерной упаковки

Специальность 05.13.18 — "Математическое моделирование, численные методы и комплексы программ"

Диссертация на соискание ученой степени кандидата физико-математических наук

Научный руководитель:
к.ф.-м.н., доцент
Ю.А. Кочетов

Новосибирск — 2010

Оглавление

Введение

Актуальность проблемы. Задачи раскроя-упаковки занимают важное место в современной комбинаторной оптимизации и привлекают внимание многих ученых как в России, так и зарубежом. Международная группа ESICUP объединяет исследователей, занимающихся задачами раскроя-упаковки, по всему миру. В настоящее время она насчитывает около пятисот участников, среди которых стоит отметить уфимскую школу под руководством профессора Э. А. Мухачевой и харьковскую школу профессора Ю. Г. Стояна.

Интерес к задачам раскроя-упаковки объясняется, в частности, их большой практической значимостью. Как правило приложения задач раскроя-упаковки относятся к материалоемким производствам, где одним из основных факторов снижения себестоимости выпускаемой продукции является рациональное использование ресурсов. На заре исследования этой проблемы Л. В. Канторовичем и В. А. Залгаллером было предложено использовать линейное программирование с неявно заданной матрицей ограничений, что позволило успешно решить важные производственные задачи. На сегодняшний день для решения задач раскроя-упаковки используются различные оптимизационные алгоритмы. В. М. Картаком и Э. А. Мухачевой разработан метод ветвей и границ для решения одномерной задачи упаковки в контейнеры. В работах Э. Х. Гимади и В. В. Залюбовского рассматриваются асимптотически точные алгоритмы. Для решения задачи гильотинной прямоугольной упаковки в контейнеры М. И. Свириденко была предложена асимптотическая полиномиальная аппроксимационная схема. Задачи раскроя-упаковки относятся к классу $\mathbb{NP}$–трудных задач комбинаторной оптимизации. Многие из них являются $\mathbb{NP}$–трудными в сильном смысле. В связи с этим большое значение приобретает разработка и исследование итерационных методов решения задач раскроя-упаковки, в том числе и ме-

тодов локального поиска, хорошо зарекомендовавших себя на практике.

Цель работы. Целью диссертационной работы является разработка и исследование методов локального поиска для решения задач раскроя-упаковки на базе эффективных алгоритмов кодирования и декодирования решений.

В соответствии с целью исследования были поставлены и выполнены следующие задачи:

1. Формулировка рассматриваемых задач раскроя-упаковки в терминах математического программирования и оценка эффективности точных методов их решения;

2. Разработка и исследование алгоритмов локального спуска и имитации отжига для решения задачи упаковки прямоугольников в прямоугольную область минимальной площади;

3. Разработка декодеров для гильотинной и негильотинной задач прямоугольной упаковки в контейнеры, позволяющих учитывать запрещенные области;

4. Разработка и исследование алгоритмов имитации отжига для решения гильотинной и негильотинной задач прямоугольной упаковки в контейнеры с запрещенными областями;

5. Разработка двухконтактного декодера для задачи упаковки кругов и прямоугольников в полосу;

6. Разработка и исследование алгоритмов вероятностного поиска с запретами для решения задачи упаковки кругов и прямоугольников в полосу;

7. Анализ эффективности разработанных методов на основе результатов численных экспериментов и сравнения с другими методами.

Методы исследования. В диссертации использованы современные методы исследования операций, включающие в себя математическое моделирование, теорию локального поиска, а также методологию экспериментальных

исследований с применением вычислительной техники и коммерческих пакетов прикладных программ для решения задач целочисленной линейной и нелинейной оптимизации.

Научная новизна работы. Для решения задачи упаковки прямоугольников в прямоугольную область минимальной площади предложен гибридный алгоритм имитации отжига, использующий новую процедуру уплотнения упаковки, аналогичную T–поздним расписаниям в календарном планировании. В ходе вычислительных экспериментов с помощью разработанного алгоритма были найдены новые рекордные значения целевой функции для семи примеров из электронных библиотек MCNC и GSRC с числом предметов от 33 до 300.

Исследован новый класс задач прямоугольной упаковки в контейнеры с запрещенными областями. Получены математические модели в терминах частично-целочисленного программирования. Для решения задач разработаны новые кодирующие схемы. На их основе разработан модифицированный алгоритм имитации отжига, позволяющий решать четыре типа задач с запрещенными областями. Экспериментально установлено, что алгоритм позволяет находить решения с малой погрешностью, в том числе и оптимальные решения на примерах с числом предметов до 20.

Для задачи упаковки кругов и прямоугольников в полосу разработана оригинальная процедура декодирования, восстанавливающая по заданной перестановке двухконтактную упаковку предметов. Разработан алгоритм вероятностного поиска с запретами с адаптивно изменяемой рандомизацией окрестности. В результате численных экспериментов для четырех известных примеров упаковки кругов в полосу получены новые рекордные значения целевой функции.

Практическая значимость работы. Предложенные в диссертационной работе алгоритмы могут быть использованы для эффективного решения следующих задач: упаковка прямоугольников в прямоугольную область минимальной площади, упаковка прямоугольников в контейнеры с запрещенными областями, упаковка кругов и прямоугольников в полосу минимальной длины. Данные задачи имеют широкий спектр практических приложений

в тех отраслях индустрии, где традиционно возникают задачи раскроя-упаковки. Разработанные алгоритмы можно использовать в практических расчетах и включать их в автоматизированные системы проектирования и управления.

Апробация работы. Полученные результаты докладывались на следующих конференциях и семинарах:

- Российская конференция «Дискретный анализ и исследование операций», г. Новосибирск, 2004;
- Международный симпозиум по исследованию операций (OR), г. Бремен, 2005, г. Карлсруэ, 2006 и г. Аугсбург, 2008;
- Всероссийская конференция «Проблемы оптимизации и экономические приложения», г. Омск, 2006;
- Азиатская международная школа-семинар «Проблемы оптимизации сложных систем», г. Новосибирск, 2006 и п. Чемал, 2008;
- Международная школа-семинар по раскрою и упаковке (ESICUP), г. Токио, 2007;
- Российская конференция «Математика в современном мире», г. Новосибирск, 2007;
- Байкальская международная школа-семинар «Методы оптимизации и их приложения», г. Северобайкальск, 2008;
- Научные семинары Института математики СО РАН.

Публикации. По теме диссертации опубликовано 11 работ, в том числе 1 статья в рецензируемом журнале из списка ВАК.

Структура и объем работы. Диссертация состоит из введения, трех глав и заключения. Объем работы составляет 104 страницы машинописного текста, включая 23 рисунка, 16 таблиц и библиографический список, содержащий 75 наименований.

Краткое содержание работы. В **первой главе** рассматривается задача упаковки конечного множества прямоугольников в прямоугольную область минимальной площади. Исследуются возможности представления решений данной задачи с помощью ориентированных деревьев. Оценивается влияние процедур уплотнения и локального спуска на относительную погрешность и время работы алгоритма имитации отжига, разработанного для решения поставленной задачи.

В разд. 1.1 содержится описание задачи. Приводится обзор уже полученных результатов. Рассматриваются основные характеристики кодирующих схем для задач прямоугольной упаковки.

В разд. 1.2 приводится математическая постановка задачи в терминах частично-целочисленного квадратичного программирования.

Разд. 1.3 содержит результаты использования коммерческого пакета GAMS для решения сформулированной задачи. Указанный пакет использует методы глобальной и локальной оптимизации и позволяет находить оптимальные решения. Указываются границы применимости такого подхода.

В разд. 1.4 и 1.5 рассматривается способ кодирования решений задачи прямоугольной упаковки с помощью ориентированных деревьев. Такая кодировка обладает линейной трудоемкостью декодирования и относительно небольшим пространством решений, что делает ее эффективной при использовании в алгоритмах локального поиска.

В разд. 1.6 определяется окрестность решений, представленных в виде ориентированных деревьев. Определяется свойство достижимости. Доказывается, что используемая окрестность обладает данным свойством.

В разд. 1.7 описываются алгоритмы локального спуска и имитации отжига, разработанные для решения поставленной задачи. Предлагается новая процедура уплотнения упаковки прямоугольников, аналогичная T–поздним расписаниям в календарном планировании. Действие процедуры заключается в поочередном смещении всех прямоугольников к нижней, правой, верхней и левой границам упаковки. Смещение реализуется за счет смены типов кодирующих деревьев. Процедура уплотнения, встроенная в алгоритмы локального поиска, меняет код решения, не меняя структуру самой упаковки. Поэтому значение целевой функции в новой точке про-

странства решений не превосходит значения в предыдущей. Использование данной процедуры в алгоритмах локального поиска открывает новые пути выхода из локальных оптимумов. Предлагается гибридный алгоритм имитации отжига с встроенной процедурой локального спуска.

В разд. 1.8 экспериментально устанавливается, что предложенная процедура уплотнения приводит к значительному сокращению погрешности и практически не влияет на время счета. Найдены новые рекордные значения целевой функции для семи примеров из электронных библиотек MCNC и GSRC. Численные эксперименты также показали, что гибридный алгоритм имитации отжига позволяет находить оптимальные решения на примерах небольшой размерности.

Во **второй главе** исследуется новая прикладная задача упаковки прямоугольников (предметов) в минимальное число контейнеров с запрещенными областями. Роль запрещенных областей могут играть как области материала низкого качества (дефекты), так и заранее размещенные предметы. Рассматриваются четыре постановки: с гильотинными разрезами, с произвольными разрезами, с поворотами предметов на 90° и без поворотов. Для решения задачи предлагается модифицированный алгоритм имитации отжига. Алгоритм использует разработанные декодеры, учитывающие запрещенные области.

В разд. 2.1 приводится содержательная постановка задачи. Описываются результаты, полученные для задач, близких к рассматриваемым.

Разд. 2.2 содержит математические постановки исследуемых задач. Задача прямоугольной упаковки с гильотинными разрезами впервые формулируется в терминах частично-целочисленного квадратичного программирования. Процесс гильотинного раскроя представлен в виде матрицы, элементам которой соответствуют прямоугольные области, полученные на этапах раскроя. Задача с произвольными разрезами формулируется в терминах частично-целочисленного линейного программирования.

В разд. 2.3 описываются способы кодирования и декодирования решений. Кодирование гильотинных упаковок осуществляется с помощью арифметических выражений в постфиксной форме, представляющих рекурсивный процесс раскроя (польские записи). Упаковки с произвольными разрезами кодируются с помощью ориентированных деревьев. Для данных ко-

дировок разработаны алгоритмы декодирования, учитывающие запрещенные области. В основе декодеров лежит жадная процедура, позволяющая на каждом шаге декодирования находить допустимое положение предмета относительно запрещенных областей.

В разд. 2.4 с использованием операций над польскими записями и ориентированными деревьями определяются окрестности линейной и квадратичной мощности соответственно. Доказывается, что введенные окрестности обладают свойством достижимости.

В разд. 2.5 приводится модифицированный алгоритм имитации отжига, осуществляющий вероятностный локальный поиск независимо для каждого контейнера. Процесс поиска идет в области допустимых и недопустимых решений, когда предметы могут выступать за границы контейнера. При каждой смене температуры осуществляется процедура уплотнения, разгружающая контейнеры, стоящие в конце списка.

В разд. 2.6 описывается жадный алгоритм построения начального решения. Принцип его работы заключается в равномерном распределении предметов среди контейнеров. Алгоритм использует нижнюю оценку, разработанную для задачи упаковки в контейнеры с запрещенными областями.

В разд. 2.7 описывается процедура уплотнения, которая позволяет концентрировать небольшие предметы в отдельных контейнерах, что облегчает их последующую разгрузку.

В разд. 2.8 приводятся результаты численных экспериментов, показывающие эффективность разработанного алгоритма при решении рассматриваемых задач и близких к ним классических задач упаковки в контейнеры. Также данный раздел содержит результаты численных экспериментов, полученных с помощью коммерческого пакета GAMS.

В **третьей главе** рассматривается задача упаковки кругов и прямоугольников в полосу минимальной длины. Для ее решения предлагается алгоритм вероятностного поиска с запретами, использующий двухконтактную схему кодирования.

В разд. 3.1 описывается содержательная постановка задачи. Приводится обзор уже полученных результатов.

В разд. 3.2 в терминах частично-целочисленного квадратичного про-

граммирования формулируется математическая постановка задачи.

В разд. 3.3 вводится понятие двухконтактной схемы кодирования: предметы упаковываются в полосу в заданном порядке так, чтобы каждый следующий предмет имел как минимум две точки касания с уже упакованными предметами, либо границами полосы. Предлагается двухконтактная кодировка с трудоемкостью декодирования $O(n^4)$ и мощностью $n!$, где n — число предметов. Доказывается теорема о том, что множество двухконтактных решений не является достаточным для нахождения глобального оптимума задачи.

В разд. 3.4 определяется окрестность квадратичной мощности.

В разд. 3.5 описывается разработанный алгоритм вероятностного поиска с запретами. Указанный алгоритм осуществляет локальный поиск по рандомизированной окрестности. В процессе поиска рандомизация окрестности адаптивно меняется в зависимости от того, как часто встречаются решения малой относительной погрешности, тем самым, осуществляя интенсификацию и диверсификацию поиска.

Разд. 3.6 содержит результаты численных экспериментов. Исследуется влияние рандомизации окрестности и длины списка запретов на работу алгоритма. Экспериментально устанавливаются значения данных параметров, при которых алгоритм наиболее эффективен. Найденные значения используются при проведении численных экспериментов на случайно сгенерированных примерах, а также на известных тестовых примерах для частных случаев рассматриваемой задачи. Для четырех известных примеров упаковки кругов в полосу удалось найти новые рекордные значения целевой функции. Также в данном разделе приводятся результаты, полученные с помощью коммерческого пакета GAMS, как для рассматриваемой задачи, так и для ее частных случаев.

Глава 1

Алгоритмы локального поиска для задачи упаковки прямоугольников в прямоугольную область минимальной площади

1.1 Введение

В задачах двумерной прямоугольной упаковки требуется без пересечений уложить на плоскости конечное множество различных предметов прямоугольной формы так, чтобы значение некоторой целевой функции достигало минимума. В качестве целевой функции могут рассматриваться площадь окаймляющего упаковку прямоугольника, длина занимаемой предметами полосы, количество используемых для упаковки контейнеров, суммарная длина взаимных связей между предметами и т. д. Задачи упаковки имеют широкое применение на практике. Например, при производстве стекла, пиломатериалов, при перевозке грузов и ведении складского хозяйства. В последнее время они стали играть значительную роль при проектировании интегральных микросхем в электронике.

В общем случае, когда допускаются произвольные движения прямоугольных предметов на плоскости, задачи упаковки практически не исследовались. Однако известны наборы предметов, у которых в оптимальной упаковке углы поворота предметов не кратны 90 градусам [33]. Гораздо более подробно исследованы ортогональные упаковки, в которых стороны предметов остаются параллельными сторонам полосы, контейнеров, либо координатным осям, а в особенности их частный случай, когда допуска-

ются только параллельные переносы заданных предметов. Далее в работе будут рассматриваться задачи ортогональной упаковки.

На сегодняшний день для решения задач указанного класса разработано множество приближенных полиномиальных алгоритмов. Например, класс уровневых алгоритмов, которые в заданном порядке размещают предметы в ряды слева направо, образуя тем самым уровни. Первым уровнем является дно полосы или контейнера. Каждый последующий уровень лежит на горизонтальной прямой, проходящей через верхнюю грань самого высокого предмета предыдущего уровня. Для задачи ортогональной прямоугольной упаковки в полосу минимальной длины примерами уровневых алгоритмов можно назвать эвристики *Следующий-Подходящий*, *Первый-Подходящий* и *Лучший-Подходящий*, являющиеся обобщением одноименных алгоритмов для одномерной задачи упаковки в контейнеры [47]. Для первых двух алгоритмов получены гарантированные оценки точности $2OPT+1$ и $\frac{17}{10}OPT+1$, в случае когда предметы упорядочены по убыванию их длин, а сами длины нормированы [25]. В работе [24] для решения задачи прямоугольной упаковки в контейнеры предлагается двухфазовый алгоритм с гарантированной оценкой точности $\frac{17}{8}OPT + 5$ при нормированных длинах предметов. Вначале с помощью уровневого алгоритма *Первый-Подходящий с упорядочением* решается двумерная задача упаковки в полосу. Затем с помощью алгоритма *Первый-Подходящий* решается одномерная задача упаковки в контейнеры. Другой известный алгоритм решения задач ортогональной упаковки называется *Нижний-Левый*. Данная эвристика не является уровневой и в соответствии с заданным порядком размещает очередной предмет, сдвигая его максимально вниз и влево. Однако в работе [14] показано, что для задачи упаковки в полосу при неудачном начальном упорядочении высота полученного с помощью алгоритма *Нижний-Левый* размещения может отличаться от оптимальной на произвольную мультипликативную константу. Тем не менее использование данного алгоритма часто встречается на практике. Например, в работе [23] предлагается его эффективная реализация для решения задачи упаковки в контейнеры за время $O(n^2)$, где n — число предметов. Для решения задачи ортогональной упаковки в заданную прямоугольную область в работе [72] предлагается полиномиальный алгоритм с трудоемкостью $O(n^5 \log n)$, моделирующий

подход человека к поставленной задаче. Подробное описание известных полиномиальных алгоритмов для задач ортогональной упаковки, а также их оценки можно найти в обзорах [2] и [56]. В работе [55] представлены результаты численных экспериментов, сравнивающих различные приближенные алгоритмы на примере задачи упаковки прямоугольников в контейнеры.

В последнее время в литературе все чаще описывается опыт применения алгоритмов локального поиска для решения задач упаковки. Важную роль при использовании такого подхода играет выбор кодирующей схемы (кодировка решений), которая определяет способ описания геометрических отношений между предметами, т. е. некоторую структуру данных для представления решений задачи и алгоритмы кодирования и декодирования. На сегодняшний день известно более десяти различных кодирующих схем. Пожалуй наиболее популярным способом представления решений для задач прямоугольной упаковки являются перестановки предметов с одним из полиномиальных алгоритмов, описанных выше, в качестве декодера. Рассмотрим кодирующие схемы, описывающие геометрические отношения для каждой пары прямоугольных предметов, т. е. их положение друг относительно друга. Алгоритм декодирования такой схемы должен восстанавливать упаковку предметов по заданным геометрическим отношениям. Для любой непересекающейся пары предметов A и B возможны следующие геометрические отношения: A выше B, A ниже B, A левее B, A правее B, A выше и левее B, A выше и правее B, A ниже и левее B, A ниже и правее B. В работе [61] представлена кодирующая схема BSG, которая с помощью таблицы задает геометрические отношения для каждой пары предметов. Чтобы закодировать решение, необходимо разместить все предметы в ячейках таблицы. Каждой ячейке может соответствовать не более одного предмета. Размер таблицы может быть произвольным. Однако множество решений, соответствующее всевозможным конфигурациям BSG–таблицы, гарантированно содержит оптимум, только при размере таблицы $n \times n$, где n — количество предметов. Преобразование таблицы в упаковку осуществляется за линейное от числа ячеек время. На основе данной кодировки в работе предлагается алгоритм имитации отжига для решения многокритериальной задачи оптимизации, возникающей при проектировании интегральных микросхем. Данная задача включает в се-

бя упаковку предметов в прямоугольную область минимальной площади с минимизацией суммарной длины взаимных связей между предметами. В работе [59] для аналогичной задачи предложен алгоритм имитации отжига на базе кодирующей схемы SP. Каждая упаковка представляется в виде пары перестановок предметов. Относительное расположение любой пары предметов можно определить, исходя из того как они располагаются друг относительно друга в перестановках. Преобразование пары перестановок в упаковку осуществляется за время $O(n^2)$. В работе [69] предложен более быстрый алгоритм декодирования с трудоемкостью $O(n \log n)$, затем его трудоемкость удалось снизить до $O(n \log \log n)$ [70]. В работе [53] предложена кодирующая схема TCG и установлено ее взаимно-однозначное соответствие с SP. Следовательно TCG имеет такое же множество решений, но в отличие от SP для представления упаковок используются два графа транзитивных замыканий, что может быть удобнее при некоторых постановках задач. На основе данной кодирующей схемы был реализован алгоритм имитации отжига для решения все той же задачи проектирования интегральных микросхем.

Кодирующие схемы BSG, SP и TCG являются достаточно гибкими и позволяют учитывать дополнительные ограничения как на сами предметы так и на область, в которой их необходимо разместить. Например, алгоритм имитации отжига в работе [61] позволяет размещать предметы в заданном многосвязном ортогональном многоугольнике. В работе [60] кодировка SP используется при решении задачи упаковки с множеством заранее размещенных предметов и предметов, не имеющих фиксированных размеров. Существенный недостаток данных кодирующих схем заключается в трудоемких процедурах декодирования, а также в большой мощности множества решений, которое для многих задач упаковки являются избыточными. В работах [22, 36] для представления компактных упаковок предлагаются кодировки $B^* - tree$ и $O - tree$, использующие двоичные и ориентированные деревья соответственно. В работах [39, 65] рассматриваются так называемые мозаичные упаковки. Для их представления используются кодирующие схемы CBL и $Q - sequence$, использующие списки топологических объектов, построенные по определенным правилам. Мощности кодировок для компактных и мозаичных решений значительно меньше, чем мощно-

сти множеств всех BSG–таблиц и пар перестановок. Кроме того декодеры $B^* - tree$, $O - tree$, CBL и $Q - sequence$ имеют линейную от числа предметов трудоемкость. Данные кодировки были использованы при реализации процедуры имитации отжига. Результаты численных экспериментов, проведенных на тестовых примерах задачи проектирования интегральных микросхем, представлены в соответствующих статьях.

Для кодирования гильотинных решений была предложена кодирующая схема NPE [71], представляющая гильотинные упаковки в виде нормированных польских записей, описывающих рекурсивный процесс раскроя. На базе кодировки NPE разработан алгоритм имитации отжига для решения задачи проектирования интегральных микросхем с условием гильотинности на размещение предметов. Также в работе установлено взаимнооднозначное соответствие между всеми нормированными польскими записями и множеством гильотинных упаковок.

Как видно, все кодирующие схемы имеют свои достоинства и недостатки. С целью выделить среди множества кодировок наиболее эффективные, в указанных работах используется понятие P–допустимой кодировки, которая должна удовлетворять следующим требованиям:

1. Пространство всех кодов конечно;

2. Каждый код допустим, т. е. ему соответствует некоторая упаковка;

3. Вычисление целевой функции для любого кода выполняется за полиномиальное время;

4. Упаковка, соответствующая коду с наилучшим значением целевой функции, является оптимальным решением задачи.

Под кодом подразумевается любое решение задачи упаковки, представленное с помощью некоторой кодирующей схемы. В табл. 1.1 указаны основные характеристики некоторых часто используемых на практике кодировок.

Заметим, что кодировка NPE не является P–допустимой только в общем случае, так как не всегда оптимальное решение задачи прямоугольной упаковки является гильотинным. Однако, при решении задач упаковки с условием гильотинности размещения предметов, которые на практике встречаются достаточно часто, использование данной кодировки дает хорошие результаты.

Таблица 1.1: Различные кодировки для задач прямоугольной упаковки

Кодирующая схема	NPE	SP	BSG	O–tree	CBL	TCG
Тип кодируемой упаковки	гильотинная	любая	любая	компактная	мозаичная	любая
P - допустимость	да в классе гильотинных решений	да	да	да	нет	да
Размер пространства решений (мощность множества всех кодов)	$O(\frac{n!2^{2.543n}}{n^{1.5}})$	$(n!)^2$	$n!C_{n^2}^n$	$O(\frac{n!2^{2n}}{n^{1.5}})$	$O(n!2^{3n})$	$(n!)^2$
Каждый код допустим	да	да	да	да	нет	да
Трудоемкость декодирования	$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n)$	$O(n)$	$O(n^2)$
Код с наилучшим значением целевой функции соответствует оптимальному решению	да в классе гильотинных решений	да	да	да при оптимизации площади упаковки	нет	да

Рассмотрим следующую задачу. Задано конечное множество прямоугольников. Каждый прямоугольник задается шириной и высотой. Требуется без пересечений разместить все прямоугольники в положительном ортанте плоскости так, чтобы площадь окаймляющего их прямоугольника была минимальна. Предполагается, что при размещении стороны прямоугольников параллельны координатным осям. Повороты прямоугольников запрещены. Ранее было доказано, что данная задача является $\mathbb{NP}$–трудной [14].

В данной главе на примере поставленной задачи исследуются возможности представления решений с помощью ориентированных деревьев (кодировка $O-tree$) [36, 68]. Данная кодировка имеет линейную трудоемкость декодирования и относительно небольшое пространство решений, достаточное для нахождения оптимальных по площади упаковок. На базе данной кодировки разработан алгоритм имитации отжига. Целью исследования является оценка влияния процедур уплотнения и локального спуска на относительную погрешность и время работы алгоритма. Разработа-

на новая процедура уплотнения упаковки прямоугольников, аналогичная Т–поздним расписаниям в календарном планировании, которая приводит к значительному сокращению погрешности и практически не влияет на время счета. Также разработан алгоритм имитации отжига, включающий в себя процедуру локального спуска, который позволяет производить поиск только среди локальных оптимумов и находить оптимальные решения на примерах небольшой размерности.

Настоящая глава организована следующим образом. В разд. 1.2 приводится математическая постановка рассматриваемой задачи. В разд. 1.3 на примере поставленной задачи исследуются возможности методов оптимизации, используемых коммерческим пакетом GAMS. В разд. 1.4 рассматривается метод представления решений задачи прямоугольной упаковки с помощью ориентированных деревьев. В разд. 1.5 описывается способ кодирования ориентированных деревьев. В разд. 1.6 определяется окрестность для решений в закодированном виде. В разд. 1.7 предлагаются алгоритмы локального поиска для решения поставленной задачи, описывается процедура уплотнения упаковки, выполняющая роль диверсификации поиска. В разд. 1.8 приводятся результаты численных экспериментов.

1.2 Математическая постановка задачи

Обозначим исходные данные задачи следующим образом: $I = \{1, 2, \ldots, n\}$ — множество прямоугольников; w_i и h_i — ширина и высота i-го прямоугольника. Введем переменные. Пусть x_i и y_i — координаты левого нижнего угла i-го прямоугольника; W и H — ширина и высота окаймляющего прямоугольника; переменная $l_{ij} \in \{0, 1\}$ равняется единице, если i-й прямоугольник находится левее j-го, и нулю в противном случае; переменная $b_{ij} \in \{0, 1\}$ равняется единице, если i-й прямоугольник находится ниже j-го, и нулю в противном случае. С использованием введенных обозначений исследуемая задача может быть представлена следующим образом:

найти $\min WH$ при ограничениях:

$$\begin{aligned} x_i &\geq 0, && i \in I, \\ y_i &\geq 0, && i \in I, \\ x_i + w_i &\leq W, && i \in I, \end{aligned}$$

$$y_i + h_i \le H, \quad i \in I.$$

Для того чтобы два прямоугольника не пересекались, достаточно чтобы один из них был левее, правее, ниже, либо выше другого. Следующая группа ограничений реализует эти условия для каждой пары прямоугольников:

$$\begin{aligned}
& x_i + w_i \le x_j + (1 - l_{ij}) \sum_{k=1}^{n} w_k, && i, j \in I, \\
& y_i + h_i \le y_j + (1 - b_{ij}) \sum_{k=1}^{n} h_k, && i, j \in I, \\
& l_{ij} + l_{ji} + b_{ij} + b_{ji} = 1, && i, j \in I.
\end{aligned}$$

В связи с тем, что в постановке задачи используются вещественные и бинарные переменные, целевая функция имеет второй порядок и все ограничения линейны, ее относят к классу задач частично-целочисленного квадратичного программирования.

1.3 Решение задачи с помощью коммерческого пакета

Задачи, записанные в терминах математического программирования, можно решать с помощью различных коммерческих пакетов, использующих методы глобальной и локальной оптимизации, например GAMS (http://www.gams.com/). Данный пакет позволяет находить оптимальные решения для ряда задач различных типов с помощью подключаемых решателей, среди которых такие известные, как BARON, CPLEX, LINDOGLOBAL, XPRESS и др. В основе решателей лежит точный алгоритм, как правило это метод ветвей и границ [10], использующий вспомогательные эвристические процедуры, позволяющие ускорить процесс поиска решения. Цель следующего эксперимента выяснить, насколько эффективно использование указанного пакета для решения сформулированной задачи. Было рассмотрено четыре тестовых примера с числом прямоугольников 33, 49, 50 и 100. Для работы на каждом примере пакету отводилось 20 часов на вычислительной машине с процессором Intel Celeron 1,8 GHz. В качестве решающего ядра для задачи, записанной в терминах частично-целочисленного квадратичного программирования, был выбран BARON (версия 8.1.1), работа которого исследовалась в [48] на примере задачи

упаковки кругов и выпуклых многоугольников. Это один из немногих решателей, способных находить глобальные оптимумы для задач частично-целочисленной нелинейной оптимизации. Однако за отведенное время с помощью указанного решателя не удалось найти ни одного допустимого решения для рассматриваемых примеров. Поэтому целевая функция исходной задачи была изменена на поиск минимального полупериметра окаймляющего прямоугольника: $\min(W + H)$. Это существенно упростило модель задачи. С линейной целевой функцией и линейными ограничениями она перешла в класс задач частично-целочисленного линейного программирования, что позволило использовать CPLEX в качестве решающего ядра. В табл. 1.2 указана площадь окаймляющего прямоугольника, нижняя оценка и оценка относительной погрешности решений, найденных с помощью CPLEX (версия 11.0). Несложно заметить, что с увеличением размерности примера эффективность пакета значительно снижается. Так при увеличении числа прямоугольников с 49 до 50 оценка относительной погрешности увеличивается на 5%, а при увеличении числа прямоугольников с 50 до 100 на 200%. Отсюда следует целесообразность разработки приближенных алгоритмов, позволяющих за небольшое время находить решения приемлемого качества для примеров указанной размерности.

Таблица 1.2: Результаты, полученные пакетом GAMS

Пример		ami33	ami49	n50	n100
MIQCP (BARON)		—	—	—	—
MIP (CPLEX)	Площадь	1314768	44754640	261232	595595
	Нижняя оценка	1156449	35445424	198579	179501
	Оценка отн. погр.	13,690%	26,263%	31,551%	231,806%

1.4 Кодирующая схема *Ориентированное дерево*

Упаковку прямоугольников будем называть L(Left)–компактной, если ни один прямоугольник невозможно сдвинуть влево при условии, что остальные прямоугольники остаются неподвижными (см. рис. 1.1 (a)). Аналогично определяются B(Bottom), R(Right) и T(Top)–компактные упаковки при невозможности сдвигов прямоугольников вниз, вправо и вверх. Если

упаковка одновременно является L–компактной и B–компактной, будем называть ее LB–компактной (см. рис. 1.1 (б)).

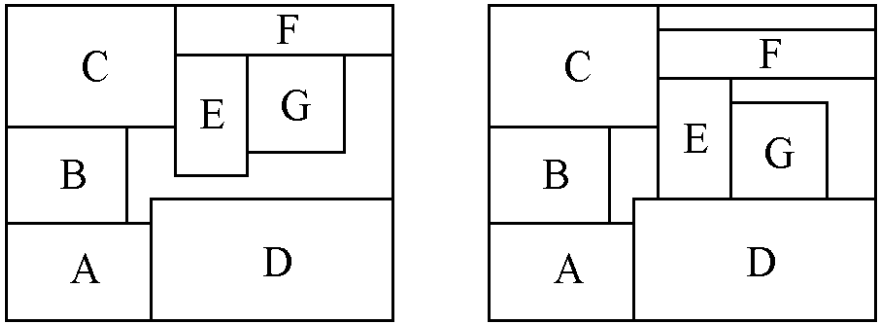


Рис. 1.1: Примеры L–компактной (а) и LB–компактной (б) упаковок

Упаковку прямоугольников можно представить с помощью ориентированного дерева, если она удовлетворяет одному из четырех введенных определений компактности. Пусть заданная упаковка является L–компактной. Тогда корнем дерева является дополнительный фиктивный прямоугольник, расположенный слева от упаковки, имеющий бесконечную высоту и нулевую ширину. Остальные вершины кодируют прямоугольники, заданные условием задачи. Ребра ставятся между вершинами, соответствующими таким прямоугольникам, которые соприкасаются по оси x и имеют пересечение проекций на ось y. В качестве родительской вершины из возможных всегда выбирается вершина, соответствующая прямоугольнику с наименьшим значением координаты y. По заданной L–компактной упаковке такое дерево можно построить за время $O(n^2)$. Будем называть его L–деревом (см. рис. 1.2 (а)).

Преобразование L–дерева в упаковку осуществляется с помощью алгоритма обхода в глубину. При этом используется дополнительная структура данных для хранения информации о текущем контуре упаковки. Контур в данном случае состоит из уже размещенных прямоугольников, покрывающих частичную упаковку сверху. При декодировании вершины дерева рассматриваются в соответствии с порядком обхода в глубину. В начале определяется x-координата прямоугольника. Ей присваивается значение $x_P + w_P$, где x_P и w_P — x-координата и ширина прямоугольника, соответствующего родительской вершине. Затем вычисляется y-координата. Для этого в контуре упаковки находятся те прямоугольники, проекции

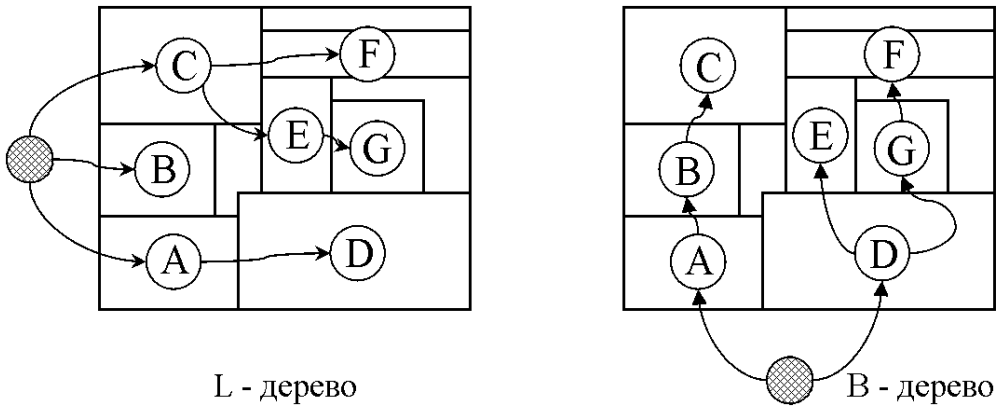


Рис. 1.2: Представление упаковки в виде L–дерева (а) и в виде B–дерева (б)

которых на ось x пресекаются с проекцией нового прямоугольника. Среди них выбирается прямоугольник, имеющий наибольшее значение суммы его y-координаты и высоты. Значение этой суммы присваивается y-координате нового прямоугольника. Другими словами, прямоугольник ставится на наивысшую точку той части контура, которая находится под ним (см. рис. 1.3). Координатам фиктивного прямоугольника, соответствующего корневой вершине, присваиваются нулевые значения. За счет использования информации о контуре текущей упаковки время работы алгоритма декодирования линейно зависит от числа прямоугольников.

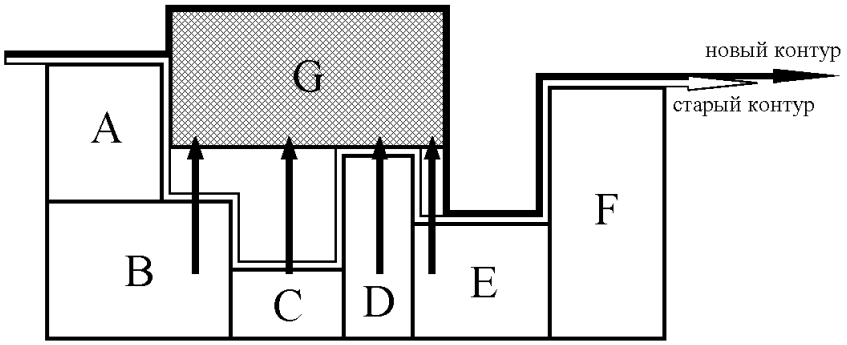


Рис. 1.3: Использование контура при декодировании

Полученная в результате декодирования L–дерева упаковка будет B–компактной, но не обязательно L–компактной. Для B–компактной упаковки аналогичным способом можно построить B–дерево (см. рис. 1.2 (б)). В отличие от L–дерева, которое имеет горизонтальную ориентацию, B–дерево будет вертикальным, а его корень будет соответствовать фиктивно-

му прямоугольнику бесконечной ширины и нулевой высоты, размещенному снизу от упаковки. При декодировании B–дерева координаты прямоугольников определяются в обратном порядке. Сначала с помощью родительской вершины вычисляется y-координата прямоугольника, а затем с помощью контура, который накрывает частичную упаковку слева, определяется x-координата. Полученная в результате декодирования B–дерева упаковка будет L–компактной.

Заданную L–компактную упаковку можно преобразовать в LB–компактную, выполняя в цикле следующую последовательность действий: кодирование в L–дерево, декодирование в B–компакт, кодирование в B–дерево, декодирование в L–компакт. Ни один шаг этого алгоритма не приводит к увеличению целевой функции. Сдвиги прямоугольников происходят либо вниз, либо влево, и следовательно настанет момент, когда ни один прямоугольник невозможно будет сдвинуть ни вниз, ни влево. Получаем, что алгоритм сходится к LB–компактной упаковке.

Заметим, что при декодировании L–дерева можно добиться смещения прямоугольников не вниз, а вверх, тогда будет получена T–компактная упаковка. Аналогично при декодировании B–дерева можно получить R–компактную упаковку вместо L–компактной. T и R–компактные упаковки кодируются с помощью T и R–деревьев, которые декодируются соответственно в L, либо R–компактную и в B, либо T–компактную упаковки. Таким образом одну и ту же упаковку можно привести к одному из четырех видов компактности и представить ее в виде соответствующего ориентированного дерева (L, B, R, либо T). Каждое из деревьев в свою очередь может быть декодировано в один из двух компактов. Далее в работе будет представлена процедура уплотнения, основанная на смене компактности упаковки, выполняющая роль диверсификации поиска в алгоритмах локального спуска и имитации отжига.

1.5 Кодирование ориентированных деревьев

Известно, что структуру ориентированного дерева можно представить с помощью двоичной последовательности и перестановки. В предыдущем разделе рассматриваются деревья, имеющие $n+1$ вершину, с фиксированным

корнем. Чтобы закодировать такое дерево, потребуется последовательность из нулей и единиц T длины $2n$ и перестановка π длины n. В перестановке хранятся вершины дерева за исключением корневой в порядке, заданном обходом в глубину, а двоичная последовательность отображает структуру дерева. Элемент «0» соответствует направлению движения при обходе, совпадающему с направлением дуги, а элемент «1» соответствует движению против направления дуги. Пример на рис. 1.4 демонстрирует описанный способ кодирования дерева.

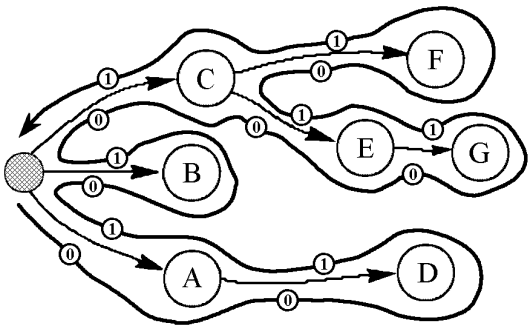


Рис. 1.4: Кодирование ориентированного дерева

Дерево имеет 8 вершин, включая корень. В закодированном виде оно будет выглядеть так: 00110100011011, $ADBCEGF$. Обход осуществляется следующим образом. Начиная с корневой вершины, идем в вершину A и записываем «0» в последовательность T и «A» в перестановку π. Затем идем в вершину D и пишем «0» в T и «D» в π. Возвращаясь к корню через вершины D и A записываем «11» в последовательность T. Далее обходим поддеревья с корневыми вершинами B и C и дописываем последовательность T и перестановку π.

Размер пространства решений данной кодирующей схемы для упаковки n прямоугольников определяется количеством всевозможных конфигураций двоичной последовательности T и множеством всех перестановок π. Из всех двоичных последовательностей рассматриваются только те, которые соответствуют структуре некоторого ориентированного дерева. Асимптотическая оценка мощности такого пространства решений равна $O\left(\frac{n!2^{2n}}{n^{1.5}}\right)$ [36].

1.6 Окрестность

Принцип работы алгоритмов локального поиска основан на понятии окрестности решения. Окрестностью текущего решения называется множество всевозможных решений, полученных путем однократного применения некоторой операции к текущему решению. Рассмотрим следующие операции над ориентированными деревьями:

(O1) Перестановка двух вершин дерева и соответствующих им прямоугольников. Данная операция меняет местами два прямоугольника в перестановке π, при этом структура ориентированного дерева не меняется. Мощность окрестности, порождаемой данной операцией, или, другими словами, количество всевозможных решений, полученных в результате применения операции к фиксированному решению, имеет порядок $O(n^2)$.

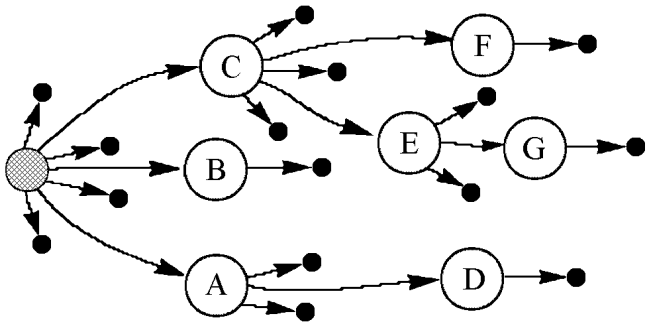


Рис. 1.5: Позиции для вставки нового листа

(O2) Перемещение листа дерева и соответствующего ему прямоугольника, в новую позицию листа. Максимально возможное число листьев у ориентированного дерева с $n+1$ вершиной равняется n. Каждому листу соответствует подпоследовательность «01» в двоичной последовательности T. После ее удаления из T количество всевозможных позиций для вставки равняется $2n-1$ (см. рис. 1.5). Таким образом, окрестность, порождаемая данной операцией, также имеет мощность порядка $O(n^2)$.

Определение *Окрестность обладает свойством достижимости, если для двух произвольных решений задачи верно, что одно может быть получено из другого за конечное число переходов к соседнему решению, определяемому с помощью данной окрестности.*

Теорема 1.1 *Окрестность, порождаемая операциями O1 и O2, обладает свойством достижимости.*

Доказательство. Легко заметить, что в рассматриваемой окрестности для существования конечного пути между двумя произвольными деревьями, недостаточно использовать только операцию O1, так как она переименовывает вершины дерева, не меняя его структуру. В то же время для получения указанного свойства достаточно использовать операцию O2, не прибегая к O1. Рассмотрим алгоритм, который, используя операцию O2, преобразует некоторое ориентированное дерево $\mathbb{T}_1$ к дереву $\mathbb{T}_2$. Так как операция O2 работает с вершинами, являющимися листьями, то преобразуем дерево $\mathbb{T}_1$ к некоторому промежуточному дереву $\widetilde{\mathbb{T}}$, в котором все вершины, соответствующие заданным прямоугольникам, являются листьями. Всего потребуется переместить не более $n-1$ вершину, если n — количество прямоугольников. Из дерева $\widetilde{\mathbb{T}}$ легко получить произвольное ориентированное дерево, в том числе и $\mathbb{T}_2$, выстраивая необходимую структуру с использованием операции O2. При этом максимальное количество вершин, которое потребуется переместить, также не превысит $n-1$. Таким образом для двух произвольных ориентированных деревьев, кодирующих некоторые упаковки, верно, что одно может быть получено из другого за не более чем $2(n-1)$ шагов по окрестности, порождаемой операцией O2. Следовательно окрестность, порождаемая операциями O1 и O2, обладает свойством достижимости. □

Данное свойство играет важную роль при выборе окрестности, так как дерево, соответствующее оптимальному решению задачи, может быть получено в результате локального поиска независимо от начального решения.

1.7 Использование алгоритмов локального поиска для решения задачи прямоугольной упаковки

Алгоритмы, в которых поиск ведется только на основании текущего решения, называются алгоритмами локального поиска. Обычно в таких алгоритмах предусматривается переход в решение, соседнее с текущим, т. е. принадлежащее его окрестности. Ранее пройденные решения не учитываются при последующих переходах. Проблема выбора окрестности достаточно важна при разработке методов локального поиска и затрагивается во многих статьях. Например, в работе [7] наряду с традиционными при-

меняют чередующиеся окрестности.

Алгоритмы локального поиска широко применяются для решения $\mathbb{NP}$–трудных задач дискретной оптимизации. Кроме того идеи локального поиска получили свое развитие в так называемых метаэвристиках [63], то есть в общих схемах построения алгоритмов, которые могут быть применены ко многим задачам дискретной оптимизации [31]. Все метаэвристики являются итерационными процедурами и для многих из них установлена асимптотическая сходимость наилучшего найденного решения к глобальному оптимуму [4]. К числу метаэвристик относятся алгоритмы имитации отжига, поиска с запретами, генетические алгоритмы, нейронные сети, муравьиные колонии и многие другие.

1.7.1 Алгоритм локального спуска

Рассмотрим алгоритм локального спуска [46]. Данный алгоритм позволяет находить локально-оптимальные решения, имеющие наилучшее значение целевой функции в заданной окрестности. Стандартный алгоритм локального спуска начинает свою работу с некоторого начального решения, выбранного случайным образом, либо с помощью вспомогательной процедуры. На каждом шаге алгоритма строится окрестность текущего решения, в которой перебором осуществляется поиск решения, лучшего, чем текущее. Если такое решение найдено, то алгоритм переходит в него и продолжает поиск, иначе работа алгоритма прекращается. Окрестность решений строится с помощью заданных операций. В нашем случае окрестностью текущего решения является множество решений, полученных однократным применением к нему либо операции O1, либо операции O2. Пусть S — некоторое решение рассматриваемой задачи, представленное в виде ориентированного дерева. Обозначим его окрестность как $N(S)$. Через $F(S)$ обозначим значение целевой функции решения S, т. е. площадь прямоугольника, окаймляющего упаковку, соответствующую дереву S. Ниже представлена схема алгоритма.

Алгоритм локального спуска

1. Построить начальное решение S.

2. Повторять, пока в окрестности $N(S)$ есть нерассмотренные решения.

   2.1. Выбрать нерассмотренное решение S' из окрестности $N(S)$.

   2.2. Если $F(S') < F(S)$, то $S := S'$.

3. Выдать результат S.

Как уже было сказано, полученное в результате выполнения алгоритма решение S будет локально-оптимальным, что означает, что в его окрестности не существует решений с лучшим значением целевой функции. Очевидно, что не каждое локально-оптимальное решение является глобально-оптимальным. Кроме того не каждый локальный оптимум имеет приемлемую относительную погрешность. В этом заключается основной недостаток алгоритма локального спуска. Попав в не очень хорошее с точки зрения значения целевой функции локально-оптимальное решение, алгоритм прекращает свой поиск. Чтобы покинуть точку локального оптимума необходимо совершить шаг ухудшающий значение целевой функции, но алгоритм локального спуска переходит только в решения лучшего качества. Рассмотрим алгоритм, позволяющий избегать подобных ситуаций.

1.7.2 Алгоритм имитации отжига

Алгоритм имитации отжига [51] представляет собой вероятностную процедуру локального поиска и принадлежит к широко известному классу алгоритмов, называемых метаэвристиками [63]. В отличие от локального спуска алгоритм имитации отжига совершает шаги как улучшающие, так и ухудшающие целевую функцию, что позволяет переходить от локальных оптимумов к другим решениям. В результате поиска поведение целевой функции становится немонотонным (см. рис. 1.6).

Процесс поиска контролируется параметром $T > 0$, который называется температурой. От данного параметра зависит вероятность, с которой алгоритм переходит в худшее решение. При фиксированной температуре вероятность сделать шаг, значительно ухудшающий текущее решение, меньше, чем шаг с небольшим ухудшением. При каждом значении температуры алгоритм выполняет заранее заданное количество шагов L, так называемая длина температурного уровня. С ростом числа итераций значение

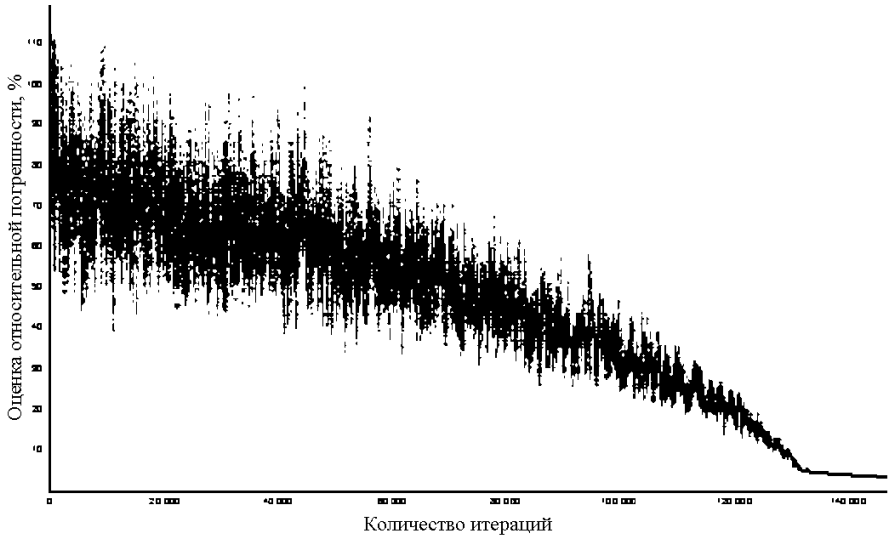


Рис. 1.6: Поведение целевой функции алгоритма имитации отжига

температуры уменьшается по закону геометрической прогрессии $T = rT$. С уменьшением температуры вероятность совершения ухудшающего шага становится меньше (см. рис. 1.7).

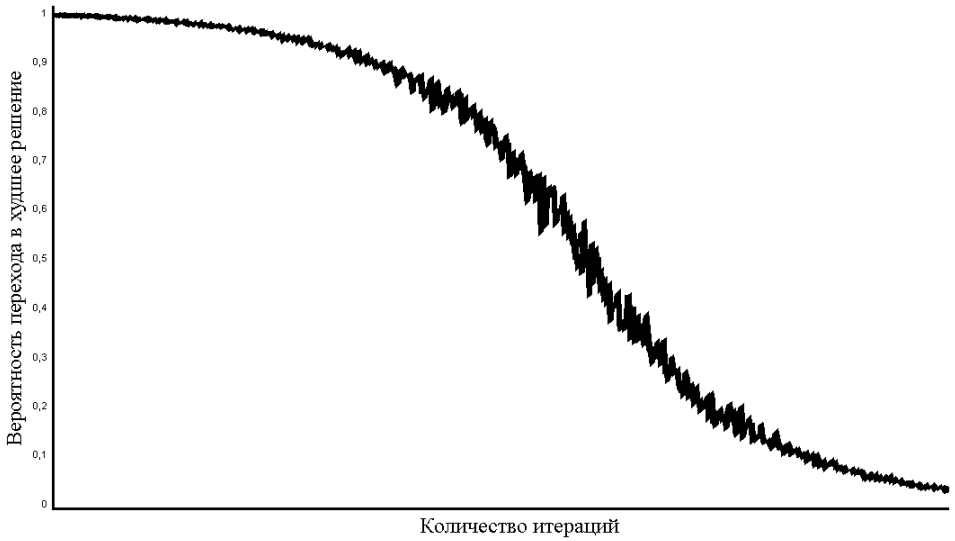


Рис. 1.7: Вероятность перехода в худшее решение

Начальное значение температуры, коэффициент охлаждения r и длина температурного уровня L подбираются экспериментально. Как правило, начальной температуре задается такое значение, чтобы на первых шагах алгоритм мог перейти к большинству решений. Коэффициент охлаждения выбирается достаточно близким к единице, обычно из диапазона $[0,8; 0,99]$. Количества шагов, совершаемых при фиксированном значении температу-

ры, должно хватать для просмотра всей окрестности. Ниже представлена схема алгоритма.

Алгоритм имитации отжига

1. Построить начальное решение S.
2. Задать начальную температуру $T > 0$.
3. Повторять, пока не выполнен критерий остановки.
   - 3.1. Выполнить цикл L раз.
     - 3.1.1. Выбрать случайным образом решение S' из окрестности $N(S)$.
     - 3.1.2. Вычислить $\Delta = F(S') - F(S)$.
     - 3.1.3. Если $\Delta \leq 0$, то положить $S := S'$.
     - 3.1.4. Если $\Delta > 0$, то положить $S := S'$ с вероятностью $e^{-\frac{\Delta}{T}}$.
   - 3.2. Понизить температуру $T = rT$.
4. Выдать лучшее найденное решение.

Критерием остановки может служить время работы алгоритма, количество выполненных итераций, либо состояние, когда на протяжении заданного числа итераций алгоритм не находит новых рекордов. Благодаря своей простоте, гибкости и эффективности алгоритм имитации отжига широко применяется на практике [31, 46]. Кроме того он обладает асимптотической сходимостью, что означает, что с ростом числа итераций вероятность оказаться в точке глобального оптимума стремится к единице, при значении температуры T стремящемся к нулю [6]. На практике нет возможности выполнить бесконечное число итераций, поэтому данное свойство является только источником вдохновения при разработке реальных алгоритмов.

1.7.3 Диверсификация поиска

Чтобы улучшить работу описанных выше алгоритмов была разработана процедура диверсификации, позволяющая в процессе поиска переходить в новые области пространства решений. Переход основан на смене компактности рассматриваемых решений с целью использовать другой тип

кодирующих деревьев. Как уже было сказано, существует четыре способа представления упаковки прямоугольников в виде ориентированного дерева. Для этого достаточно привести упаковку к одному из четырех компактных видов и затем закодировать ее с помощью соответствующего дерева (L, B, R, либо T). Каждое дерево можно декодировать двумя способами. При декодировании горизонтального дерева (L и R) можно получить либо B–компактную, либо T–компактную упаковку. Вертикальное дерево (B и T) декодируется либо в L–компактную, либо в R–компактную упаковку. Будем использовать декодеры, получающие следующие упаковки:

$$\text{L–дерево} \Longrightarrow \text{B–компактная упаковка}$$
$$\text{B–дерево} \Longrightarrow \text{R–компактная упаковка}$$
$$\text{R–дерево} \Longrightarrow \text{T–компактная упаковка}$$
$$\text{T–дерево} \Longrightarrow \text{L–компактная упаковка}$$

Процедура диверсификации работает следующим образом. Если алгоритм в данный момент использует L–деревья, то текущее решение декодируется в B–компакт и затем кодируется в B–дерево. После чего алгоритм работает с B–деревьями. Если используются B–деревья, то алгоритм переходит к использованию R–деревьев. R–деревья заменяются на T–деревья. И, наконец, T–деревья заменяются на L–деревья. Смена типа кодирующих деревьев в таком порядке позволяет не тратить время на приведение упаковки к соответствующему компактному виду, так как в результате декодирования упаковка уже обладает требуемой компактностью.

Данная процедура применяется в описанных алгоритмах локального поиска следующим образом. Алгоритм локального спуска находит локально-оптимальную упаковку с использованием L–деревьев. Затем кодирует ее в B–дерево и продолжает спуск. Полученный локальный оптимум кодируется в R–дерево. И так далее по циклу L→B→R→T→L до тех пор, пока полученное решение не будет локально-оптимальным независимо от того, какой тип кодирующих деревьев используется. Алгоритм имитации отжига использует процедуру диверсификации каждый раз при смене температуры в качестве дополнительного шага 3.3 общей схемы, осуществляя кодирование текущего решения в следующий тип дерева по правилу L→B→R→T→L. Влияние, оказываемое разработанной процедурой, на качество получаемых решений исследуется в разделе численных экспериментов.

1.7.4 Гибридный алгоритм

Алгоритм имитации отжига имеет значительное преимущество перед локальным спуском за счет того, что позволяет переходить от одного локального оптимума к другому. Но при этом ему приходится рассматривать множество не оптимальных решений, как правило, гораздо большее, чем локальному спуску при поиске локального оптимума из произвольной начальной точки. В работе [58] на примере задач коммивояжера и разбиении графа были изучены возможности алгоритма имитации отжига со встроенной процедурой локального спуска. Такая реализация алгоритма позволяет выполнять поиск только среди локальных оптимумов. В отличие от стандартной схемы отжига данному алгоритму не приходится решать идти или не идти в точку не являющуюся локальным оптимумом, и выбор сводится только к следующему — остаться в текущем локальном оптимуме или перейти к следующему. Кроме того, алгоритм имеет существенное преимущество перед локальным спуском, унаследовав асимптотическую точность от стандартной схемы отжига.

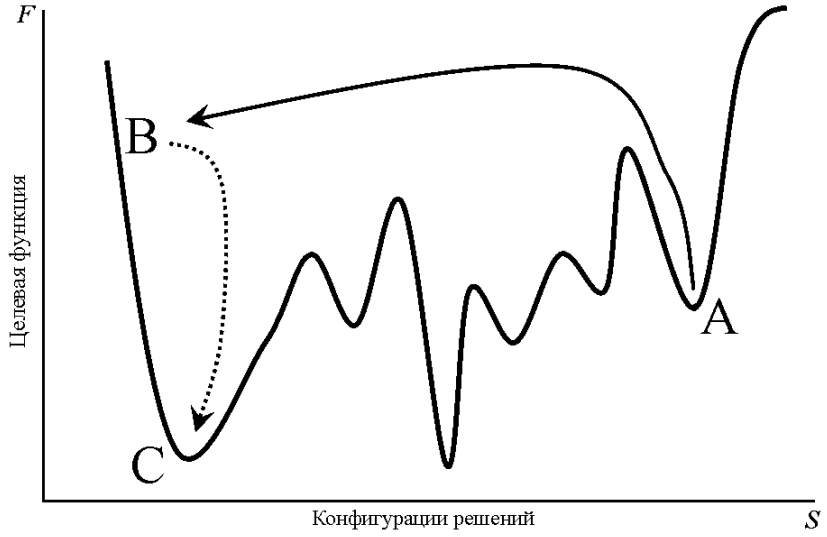


Рис. 1.8: Принцип работы гибридного алгоритма имитации отжига

Предположим, что алгоритм уже находится в некотором локально-оптимальном решении (точка A на рис. 1.8). Применим операцию, существенно меняющую текущее решение, в результате чего получим некоторое промежуточное решение (точка B). Стандартный алгоритм имитации отжига проводит испытание, чтобы решить — остаться в этой точке или вер-

нуться к предыдущей. Гибридный алгоритм сначала оптимизирует полученное решение с помощью локального спуска и получает новый локальный оптимум (точка C), а затем решает перейти в него или нет. Такой подход позволяет преодолевать за один шаг несколько «гребней» целевой функции в отличие от стандартного алгоритма имитации отжига, который «взбирается» на каждую вершину по отдельности. В данной работе гибридный алгоритм реализован следующим образом.

Гибридный алгоритм имитации отжига

1. Построить начальное решение S_0.
2. Выполнить локальный спуск и получить решение S.
3. Задать начальную температуру $T := T_0$.
4. Повторять, пока не выполнен критерий остановки.
   - 4.1. Выполнить цикл L раз.
     - 4.1.1. Применить к решению S случайное количество (от 1 до 10) операций O1 и O2 и получить решение S'_0.
     - 4.1.2. Выполнить локальный спуск и получить решение S'.
     - 4.1.3. Вычислить $\Delta = F(S') - F(S)$.
     - 4.1.4. Если $\Delta \leq 0$, то положить $S := S'$.
     - 4.1.5. Если $\Delta > 0$, то положить $S := S'$ с вероятностью $e^{-\frac{\Delta}{T}}$.
   - 4.2. Понизить температуру $T = rT$.
   - 4.3. Выполнить процедуру диверсификации.
5. Выдать лучшее найденное решение.

1.8 Численные эксперименты

В данном разделе проводится анализ работы алгоритмов локального спуска и имитации отжига на примере рассматриваемой задачи прямоугольной упаковки. На языке PASCAL были реализованы как стандартные схемы

данных алгоритмов, так и схемы с использованием разработанной процедуры диверсификации. Тестирование проводилось на вычислительной машине с процессором AMD Athlon 1,44 GHz. Были рассмотрены 4 тестовых примера из известных электронных библиотек MCNC и GSRC (http://vlsicad.eecs.umich.edu/BK/FPUtils/). Данные библиотеки используются во многих работах, посвященных задачам упаковки, возникающим в проектировании интегральных микросхем [36, 39, 53, 60, 69, 73].

Таблица 1.3: Результаты, полученные алгоритмами локального поиска

	Библиотека	MCNC	MCNC	GSRC	GSRC
	Пример	ami33 ($n = 33$)	ami49 ($n = 49$)	n50 ($n = 50$)	n100 ($n = 100$)
LS	Лучшее	10,091%	12,377%	15,055%	13,269%
	Худшее	31,840%	37,021%	39,457%	31,954%
	Среднее	21,000%	22,115%	21,328%	21,432%
	Время работы	10мс	50мс	80мс	7с
	LB–компакты	38 из 100	29 из 100	20 из 100	17 из 100
LS*	Лучшее	9,467%	7,255 %	10,908%	11,322%
	Худшее	15,348%	15,471%	13,956%	16,230%
	Среднее	12,794%	11,273%	12,395%	13,974%
	Время работы	40мс	70мс	2с	10с
	LB–компакты	100 из 100	100 из 100	100 из 100	100 из 100
SA	Лучшее	2,829%	3,284 %	3,079%	2,591%
	Худшее	8,897%	12,674%	7,961%	6,832%
	Среднее	5,530%	5,787 %	5,555%	4,457%
	Время работы	18с	45с	5с	17с
	LB–компакты	97 из 100	89 из 100	97 из 100	95 из 100
SA*	Лучшее	1,948%	1,818%	2,069%	2,312%
	Худшее	6,055%	4,273%	4,807%	5,538%
	Среднее	3,761%	3,086%	3,557%	3,514%
	Время работы	21с	40с	8с	20с
	LB–компакты	100 из 100	100 из 100	100 из 100	100 из 100

1.8.1 Влияние диверсификации поиска

Цель эксперимента заключалась в том, чтобы сравнить эффективность работы двух алгоритмов: локального спуска LS и имитации отжига SA. А также выяснить, насколько существенное влияние оказывает на них использование процедуры диверсификации поиска (алгоритмы LS* и SA*). Для

каждого алгоритма было выполнено 400 запусков, по 100 запусков на каждый пример. Поиск начинался с некоторого случайного решения. Начальная температура имитации отжига бралась равной $\max\limits_{S' \in N(S)} |F(S) - F(S')|$. Коэффициент охлаждения $r = 0,95$. Длина температурного уровня равнялась мощности окрестности, т. е. $L = n^2$. Критерий остановки срабатывал, когда процесс поиска «остывал». Это выражалось в том, что на протяжении пяти температурных уровней подряд было принято менее 10% рассмотренных решений.

В таблице 1.3 представлены результаты вычислений. Вместо площади окаймляющего упаковку прямоугольника указана оценка относительной погрешности:

$$D = \left(1 - \frac{\sum\limits_{i=1}^{n} w_i h_i}{WH} \right) \cdot 100\%.$$

Для каждого примера в таблице указаны лучшая/худшая/средняя оценка относительной погрешности, среднее время работы алгоритма, а также сколько полученных решений являются LB–компактными.

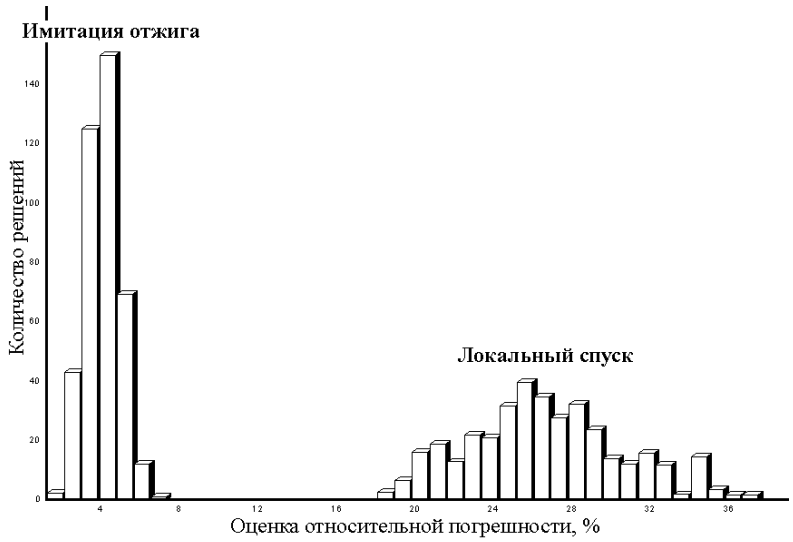


Рис. 1.9: Распределение решений, полученных локальным спуском и имитацией отжига

На диаграмме, изображенной на рис. 1.9, показано, как распределились решения, полученные с помощью локального спуска и имитации отжига. Возможность покидать локальные оптимумы дает алгоритму имитации отжига преимущество перед локальным спуском, в результате чего погреш-

ность решений, получаемых с ее помощью оказывается значительно меньше.

Существенное влияние на работу алгоритмов оказала диверсификация поиска (см. рис. 1.10). Предложенная процедура меняет код решения, не меняя структуру самой упаковки. Поэтому значение целевой функции в новой точке не превосходит значения в предыдущей, и поиск продолжается дальше. При этом открываются новые области для поиска решений, и появляются новые пути выхода из локальных оптимумов, что приводит к лучшим результатам. Как видно из табл. 1.3 использование данной процедуры не приводит к существенному увеличению времени работы алгоритмов.

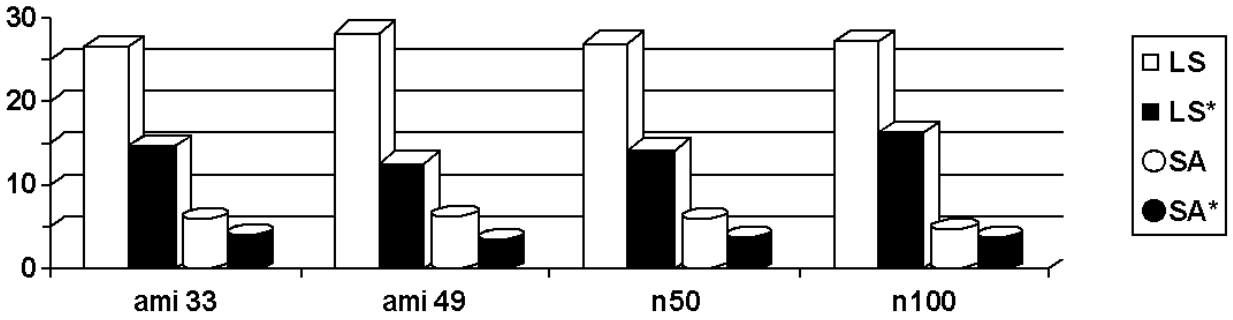


Рис. 1.10: Влияние процедуры диверсификации на работу алгоритмов

С помощью алгоритма SA* были найдены новые рекордные значения целевых функций для ряда примеров из библиотек MCNC и GSRC. В табл. 1.4 приводятся наилучшие известные результаты и результаты, полученные разработанным алгоритмом имитации отжига с процедурой уплотнения упаковки.

Таблица 1.4: Новые рекордные решения примеров из библиотек MCNC и GSRC

Библиотека	MCNC	MCNC	GSRC	GSRC	GSRC	GSRC	GSRC
Пример	ami33	ami49	n100	n100b	n200	n200b	n300
Размерность	$n = 33$	$n = 49$	$n = 100$	$n = 100$	$n = 200$	$n = 200$	$n = 300$
Пред. результат	1,996%	2,300%	3,849%	3,602%	5,504%	6,290%	6,800%
SA*	1,948%	1,818%	2,312%	2,243%	2,966%	3,886%	4,993%

В таблице указаны значения оценки относительной погрешности. Лучшие значения целевых функций для примеров ami33 и ami49 были ранее получены с помощью алгоритма имитации отжига, описанного в работе

[52]. Предыдущее рекордное решение примера n300 было получено с помощью алгоритма, представленного в работе [21]. В этой же работе были найдены решения других четырех примеров, полученные с использованием кодирующей схемы, описанной в [22].

1.8.2 Гибридный алгоритм с диверсификацией поиска

Следующий эксперимент демонстрирует возможности гибридного алгоритма имитации отжига SA(LS)*. Вычисления показали, что данный алгоритм является эффективным и позволяет находить точные решения для задач малых размерностей. Но при этом его трудоемкость значительно возрастает с ростом числа прямоугольников. Поэтому теряется смысл использовать его уже при $n = 20$. Возникает необходимость искать способы сократить время выполнения за счет других характеристик, например, качества решений. Но цель данной работы не включала в себя рассмотрение этих вопросов.

Таблица 1.5: Нахождение оптимальных решений

Число прямоугольников		$n = 10$	$n = 12$	$n = 14$	$n = 15$
SA*	Оптимальные решения	56%	23%	7%	1%
	Время работы	10мс	12мс	9мс	7мс
SA(LS)*	Оптимальные решения	100%	100%	100%	100%
	Время работы	80мс	21с	105с	270с

В табл. 1.5 показана частота нахождения точных решений алгоритмами SA* и SA(LS)*, а также среднее время работы до срабатывания критерия остановки. В эксперименте рассматриваются примеры с 10, 12, 14 и 15 прямоугольниками. Для каждой размерности было сгенерировано 100 примеров. Генерация заключалась в разрезании случайным образом квадрата 100×100 на заданное количество прямоугольников. Таким образом, оптимальное решение для каждого примера известно. Алгоритмы работали с одинаковыми начальными параметрами, заданными в предыдущем эксперименте.

1.9 Выводы к главе 1

Рассмотрена задача упаковки прямоугольников в прямоугольную область минимальной площади. Проведен обзор кодировок, использующихся для задач данного типа. На основе кодирующей схемы *Ориентированное дерево* разработаны алгоритмы локального спуска, имитации отжига, а также алгоритм имитации отжига, комбинированный с локальным спуском. Проанализированы основные особенности предложенных алгоритмов. Приведено подробное описание их реализации для решения поставленной задачи, а также исследованы их возможности как на известных тестовых примерах, так и на сгенерированных случайным образом. Разработана процедура уплотнения упаковки, использующая возможности кодировки *Ориентированное дерево*. Показано, что использование данной процедуры в качестве диверсификации поиска приводит к уменьшению относительной погрешности получаемых решений и практически не влияет на время работы алгоритмов. В результате численных экспериментов были найдены новые рекордные значения целевой функции для семи примеров из библиотек MCNC и GSRC, что позволяет говорить о высокой эффективности разработанных методов локального поиска для решения задачи прямоугольной упаковки.

Глава 2

Алгоритм имитации отжига для задач прямоугольной упаковки в контейнеры с запрещенными областями

2.1 Введение

Рассматривается следующая оптимизационная задача. Задан конечный набор прямоугольных предметов. Каждый предмет имеет свою ширину и высоту. Имеется конечный список прямоугольных контейнеров. Размеры контейнеров заданы. Каждый контейнер имеет конечное число запрещенных областей прямоугольной формы с фиксированными координатами. Каждый предмет имеет свойство, позволяющее ему пересекаться с запрещенными областями, либо нет. Кроме того, имеется достаточное количество контейнеров без запрещенных областей с одинаковыми размерами. Требуется с учетом запрещенных областей разместить все предметы в минимальном числе контейнеров. При этом учитывается порядок, заданный на множестве контейнеров. Сначала заполняется первый контейнер, затем, если необходимо, второй и т. д. Если все предметы не удается разместить в заданном списке контейнеров с запрещенными областями, то используются дополнительные контейнеры без запрещенных областей. Предполагается, что при размещении предметы не пересекаются друг с другом, и стороны предметов параллельны сторонам контейнеров. Данная задача является обобщением хорошо известной двумерной задачи упаковки в контейнеры и $\mathbb{NP}$–трудна в сильном смысле [55].

В настоящей главе рассматриваются четыре варианта поставленной за-

дачи двумерной упаковки в контейнеры (2BP): ориентированная задача (O) — повороты предметов не допускаются; неориентированная задача (R) — допускаются повороты предметов на 90°; гильотинная задача (G) — каждый предмет может быть вырезан из контейнера при помощи рекурсивных разрезов от края до края; и негильотинная задача(F) — допускаются произвольные размещения предметов в контейнерах. На рис. 2.1 указана связь между различными постановками. Как видно, неориентированная задача с произвольными разрезами 2BP|R|F является наиболее общей.

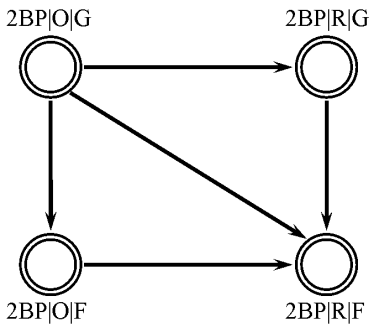


Рис. 2.1: Классификация задач упаковки в контейнеры

Поставленная задача имеет прикладной характер и относится к классу задач двумерного раскроя и упаковки. Из известных задач наиболее близкими к рассматриваемой задаче являются задачи упаковки различных прямоугольных предметов в минимальное число одинаковых прямоугольных контейнеров и задачи прямоугольной упаковки с множеством заранее размещенных предметов, возникающие при проектировании интегральных микросхем. Различные варианты задачи прямоугольной упаковки в контейнеры и алгоритмы ее решения приводятся в работах [17, 18, 55]. В работе [17] рассматривается ориентированный случай задачи и предлагаются новые нижние оценки оптимума. В работе [18] рассматривается задача упаковки в контейнеры множества прямоугольников, когда некоторые из них можно поворачивать на 90 градусов, а некоторые нет. Ориентированная и неориентированная задачи прямоугольной упаковки в контейнеры являются частными случаями данной задачи. В работе предлагаются нижние оценки оптимума и приближенный метод решения. Четыре возможных случая задачи (ориентированная/неориентированная, гильотинные/произвольные разрезы) без запрещенных областей рассматриваются в работе [55]. Для решения предлагается алгоритм поиска с запретами, ис-

пользующий в зависимости от типа решаемой задачи различные декодеры. Задачи прямоугольной упаковки в контейнеры различных размеров рассматриваются в работах [62, 64]. В первой работе предлагается несколько приближенных уровневых алгоритмов. Во второй работе рассматривается задача упаковки в контейнеры, имеющие различные размеры и стоимость. Предлагается математическая модель в терминах целочисленного линейного программирования и точный метод для нахождения упаковки минимальной стоимости.

Если ни один предмет не может пересекаться с запрещенными областями, то сами запрещенные области могут рассматриваться как заранее размещенные предметы. Задачи прямоугольной упаковки с такими условиями возникают при проектировании интегральных микросхем, когда расположение некоторых компонентов будущей схемы уже известно. Наиболее часто здесь встречается задача прямоугольной упаковки в угол, где необходимо минимизировать площадь окаймляющего упаковку прямоугольника и длину связей между компонентами, составляющими упаковку. В работе [28] для решения задачи прямоугольной упаковки предметов в положительный ортант плоскости с непустым множеством заранее размещенных предметов предлагается жадный алгоритм с трудоемкостью $O(n^5 log\ n)$. В работе [60] рассматривается похожая задача, но наряду с уже размещенными предметами задается множество предметов не имеющих фиксированных размеров, а имеющих фиксированную площадь. Для решения данной задачи используется алгоритм имитации отжига, представляющий упаковку предметов в виде пары перестановок. Эта же задача, но с условием гильотинных разрезов решается в работе [73] также с помощью имитации отжига. В работе [27] задача прямоугольной упаковки с заранее размещенными предметами решается с помощью релаксаций и штрафных функций. В работе [3] для решения задач прямоугольной упаковки в полосу и в контейнеры с дефектами материала предлагается алгоритм имитации отжига с блочным декодером, рассматривающим дефекты как детали с фиксированным расположением.

В настоящей главе предлагается модифицированный алгоритм имитации отжига для задач прямоугольной упаковки в контейнеры с запрещенными областями. Решения представляются с помощью польских записей

в задаче с гильотинными разрезами, и с помощью ориентированных деревьев в задаче с произвольными разрезами. Для данных кодирующих схем разработаны декодеры, учитывающие запрещенные области при построении упаковок. В процессе работы алгоритма имитации отжига регулярно выполняется процедура уплотнения, которая перемещает предметы из контейнеров стоящих в конце списка в контейнеры, находящиеся ближе к началу списка.

Глава организована следующим образом. В разд. 2.2 приводятся математические постановки задач. В разд. 2.3 описываются кодирующие схемы. В разд. 2.4 определяются окрестности решений. В разд. 2.5 приводится схема модифицированного алгоритма имитации отжига для рассматриваемых типов поставленной задачи. В разд. 2.6 и 2.7 описываются алгоритмы построения начального решения и уплотнения упаковки. В разд. 2.8 приводятся результаты численных экспериментов.

2.2 Математические постановки задач

В данном разделе приводятся математические постановки задач прямоугольной упаковки в контейнеры с запрещенными областями для произвольных и гильотинных разрезов.

Теорема 2.1 *Задачи 2BP|O|F и 2BP|R|F с запрещенными областями могут быть сформулированы в терминах частично-целочисленного линейного программирования.*

Доказательство. Введем следующие обозначения: $I = \{1, 2, \ldots, n\}$ — множество предметов; w_i и h_i, $i \in I$, — ширина и высота i-го предмета; d_i — параметр, принимающий значение 1, если i-й предмет может пересекаться с запрещенными областями, и значение 0 в противном случае. Пусть $M = M_1 \cup M_2$ — множество контейнеров, где $M_1 = \{1, 2, \ldots, l\}$ — контейнеры с запрещенными областями, а $M_2 = \{1, 2, \ldots, n\}$ — дополнительные контейнеры одинакового размера без запрещенных областей. Пусть W_m и H_m, $m \in M$, — ширина и высота m-го контейнера. Предполагается, что ширина и высота дополнительных контейнеров не меньше максимальной ширины и высоты предметов. Поэтому достаточно n дополнительных контейнеров для существования допустимого решения задачи. Будем считать,

что левый нижний угол каждого контейнера находится в начале координат. Обозначим множество запрещенных областей как $I^0 = \{1, 2, \ldots, K\}$, а их координаты и размеры как $x_k^0,\ y_k^0$ и $w_k^0,\ h_k^0,\ k \in I^0$. Параметр q_{km} равняется единице, если k-я запрещенная область находится в m-м контейнере, и нулю в противном случае.

Введем переменные задачи. Пусть x_i и y_i — координаты левого нижнего угла i-го предмета. Переменная $z_i \in \{0, 1\}$ принимает значение 1, если i-й предмет повернут на 90 градусов, и значение 0 в противном случае. Пусть бинарные переменные $a_{ij}^L,\ a_{ij}^B,\ i, j \in I$, равняются единице, если i-й предмет находится левее, либо соответственно ниже j-го предмета, и нулю в противном случае. Аналогично бинарные переменные $b_{ik}^L,\ b_{ik}^R,\ b_{ik}^B,\ b_{ik}^A,\ i \in I,\ k \in I^0$, принимают значение, равное единице, если i-й предмет находится левее, правее, ниже, либо выше k-й запрещенной области, и значение ноль в противном случае. Переменные $p_{im} \in \{0, 1\},\ i \in I,\ m \in M$, определяют набор предметов, содержащийся в каждом контейнере, следующим образом: p_{im} равняется единице, если i-й предмет лежит в m-м контейнере, и нулю в противном случае. Переменная $u_m \in \{0, 1\}$ принимает значение 1, если m-й контейнер используется, и значение 0 иначе. Тогда рассматриваемая задача может быть представлена следующим образом

$$\min \sum_{m=1}^{l+n} u_m.$$

Зададим порядок использования контейнеров от первого к последнему:

$$u_m \geq u_{m+1},\ m \in M.$$

Использование контейнеров определяется с помощью следующего неравенства:

$$u_m n \geq \sum_{i=1}^{n} p_{im},\ m \in M.$$

Следующее условие гарантирует, что каждый предмет находится только в одном контейнере:

$$\sum_{m=1}^{l+n} p_{im} = 1,\ i \in I.$$

Выпишем условия, гарантирующие, что все предметы будут размещены в пределах используемого контейнера:

$$
\begin{aligned}
& x_i \geq 0, && i \in I, \\
& y_i \geq 0, && i \in I, \\
& x_i + w_i(1 - z_i) + h_i z_i \leq \sum_{m=1}^{l+n} p_{im} W_m, && i \in I, \\
& y_i + h_i(1 - z_i) + w_i z_i \leq \sum_{m=1}^{l+n} p_{im} H_m, && i \in I.
\end{aligned}
$$

Следующие три ограничения исключают пересечения предметов между собой, если они находятся в одном контейнере. Здесь W_{max} и H_{max} — максимальная ширина и максимальная высота контейнеров:

$$
\begin{aligned}
& x_i + w_i(1 - z_i) + h_i z_i \leq x_j + (1 - a_{ij}^L) W_{max}, && i, j \in I, \\
& y_i + h_i(1 - z_i) + w_i z_i \leq y_j + (1 - a_{ij}^B) H_{max}, && i, j \in I, \\
& a_{ij}^L + a_{ji}^L + a_{ij}^B + a_{ji}^B \geq p_{im} + p_{jm} - 1, && i, j \in I,\ m \in M.
\end{aligned}
$$

Аналогично последняя группа неравенств исключает пересечения предметов и запрещенных областей, находящихся в одном контейнере в случаях, когда это недопустимо:

$$
\begin{aligned}
& x_i + w_i(1 - z_i) + h_i z_i \leq x_k^0 + \left(1 - b_{ik}^L\right) W_{max}, && i \in I,\ k \in I^0, \\
& x_k^0 + w_k^0 \leq x_i + \left(1 - b_{ik}^R\right) W_{max}, && i \in I,\ k \in I^0, \\
& y_i + h_i(1 - z_i) + w_i z_i \leq y_k^0 + \left(1 - b_{ik}^B\right) H_{max}, && i \in I,\ k \in I^0, \\
& y_k^0 + h_k^0 \leq y_i + \left(1 - b_{ik}^A\right) H_{max}, && i \in I,\ k \in I^0, \\
& b_{ik}^L + b_{ik}^R + b_{ik}^B + b_{ik}^A \geq p_{im} + q_{km} - d_i - 1, && i \in I,\ k \in I^0,\ m \in M.
\end{aligned}
$$

Выписанные условия определяют неориентированную задачу с произвольными разрезами 2BP|R|F. Аналогично при $z_i \equiv 0$ формулируется ориентированная задача 2BP|O|F. $\square$

Теорема 2.2 *Задачи 2BP|O|G и 2BP|R|G с запрещенными областями могут быть сформулированы в терминах частично-целочисленного квадратичного программирования с линейной целевой функцией.*

Доказательство. Чтобы представить рекурсивный процесс разрезания прямоугольного контейнера на $\bar{n}$ предметов в терминах математического программирования, рассмотрим матрицу $\bar{n} \times \bar{n}$, изображенную на рис. 2.2.

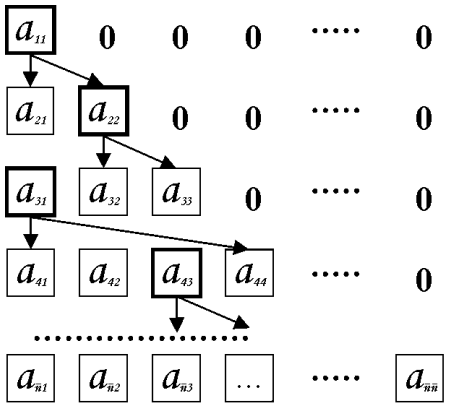


Рис. 2.2: Матрица гильотинного раскроя

Все элементы матрицы над главной диагональю тождественно равны нулю. Каждому ненулевому элементу соответствует некоторая прямоугольная область, имеющая свои координаты и размеры. Элементу a_{11} соответствует сам прямоугольный контейнер. Процесс гильотинного раскроя можно представить в виде последовательности переходов от первой строки матрицы ко второй, третьей и т. д. При каждом переходе совершается разрез одной из областей предыдущей строки на две части. Одна часть сохраняется в следующей строке на месте выбранной области, другая добавляется в качестве новой, увеличивая количество элементов-областей на единицу. Чтобы получить $\bar{n}$ предметов, необходимо сделать $\bar{n} - 1$ разрез. В последней строке получаем $\bar{n}$ областей $a_{\bar{n}1}, \ldots, a_{\bar{n}\bar{n}}$, содержащих в себе ровно один прямоугольный предмет.

Используем описанный способ для представления гильотинного раскроя нескольких контейнеров. Каждому контейнеру будет соответствовать своя матрица гильотинного раскроя. Так как заранее неизвестно, сколько предметов будет содержаться в каждом контейнере, то размер матриц будет максимально возможным — $n \times n$.

Обозначим исходные данные задачи и переменные x_i, y_i, z_i, b^L_{ik}, b^B_{ik}, b^R_{ik}, b^A_{ik}, u_m, аналогично предыдущей теореме. Введем также следующие переменные: $\overline{X}^m_{ij}$, $\overline{Y}^m_{ij}$, $\overline{W}^m_{ij}$, $\overline{H}^m_{ij}$, $i, j \in I$, $m \in M$, — координаты и размеры прямоугольных областей, соответствующих элементам матриц гильотинного раскроя. Переменная $g^m_{ij} \in \{0, 1\}$, $1 \le j < i \le n - 1$, $m \in M$, принимает значение 1, если при раскрое m-го контейнера i-м разрезом разрезается j-я область, при этом разрез горизонтальный, и значение 0 в остальных

случаях. Аналогично введем переменную $v_{ij}^m \in \{0,1\}$ для вертикальных разрезов. Переменная s_i^m принимает значение, равное длине отрезаемой части i-м разрезом при раскрое m-го контейнера. Для привязки предметов к контейнерам будем использовать переменные $r_{ij}^m \in \{0,1\}$, $i,j \in I$, $m \in M$, принимающие значение 1, если i-й предмет в m-м контейнере занимает j-ю область, и значение 0 в противном случае. Тогда математическая постановка задачи прямоугольной упаковки в контейнеры с запрещенными областями в гильотинном случае будет иметь следующий вид

$$\min \sum_{m=1}^{l+n} u_m.$$

Условие, задающее порядок контейнеров остается прежним:

$$u_m \geq u_{m+1},\ m \in M.$$

Контейнер используется, если в нем содержится хотя бы один предмет:

$$u_m n \geq \sum_{i,j=1}^{n} r_{ij}^m,\ m \in M.$$

Каждый предмет находится только в одной области одного из контейнеров:

$$\sum_{m=1}^{l+n} \sum_{j=1}^{n} r_{ij}^m = 1,\ i \in I.$$

В каждой области находится не более одного предмета:

$$\sum_{i=1}^{n} r_{ij}^m \leq 1,\ j \in I,\ m \in M.$$

Элементам a_{11} матриц гильотинного раскроя присваиваются параметры контейнеров:

$$\begin{aligned}
&\overline{X}_{11}^m = 0, && m \in M,\\
&\overline{Y}_{11}^m = 0, && m \in M,\\
&\overline{W}_{11}^m = u_m W_m, && m \in M,\\
&\overline{H}_{11}^m = u_m H_m, && m \in M.
\end{aligned}$$

Выполнение следующего условия гарантирует, что при переходе на новую строку в матрице гильотинного раскроя совершается не более одного разреза:

$$\sum_{j=1}^{i} \left(g_{ij}^m + v_{ij}^m\right) \le u_m, \qquad 1 \le i \le n-1,\ m \in M.$$

После каждого разреза координаты и размеры прямоугольных областей преобразуются следующим образом:

$$\begin{aligned}
&\overline{X}_{i+1,j}^m = \overline{X}_{ij}^m, && 1 \le j < i \le n-1,\ m \in M,\\
&\overline{Y}_{i+1,j}^m = \overline{Y}_{ij}^m, && 1 \le j < i \le n-1,\ m \in M,\\
&\overline{W}_{i+1,j}^m = \overline{W}_{ij}^m - v_{ij}^m s_i^m, && 1 \le j < i \le n-1,\ m \in M,\\
&\overline{H}_{i+1,j}^m = \overline{H}_{ij}^m - g_{ij}^m s_i^m, && 1 \le j < i \le n-1,\ m \in M.
\end{aligned}$$

В результате разреза добавляется новая область:

$$\overline{X}_{i+1,i+1}^m = \sum_{j=1}^{i} \left(\left(g_{ij}^m + v_{ij}^m\right)\overline{X}_{ij}^m + v_{ij}^m \overline{W}_{i+1,j}^m\right), \quad 1 \le i \le n-1,\ m \in M,$$

$$\overline{Y}_{i+1,i+1}^m = \sum_{j=1}^{i} \left(\left(g_{ij}^m + v_{ij}^m\right)\overline{Y}_{ij}^m + g_{ij}^m \overline{H}_{i+1,j}^m\right), \quad 1 \le i \le n-1,\ m \in M,$$

$$\overline{W}_{i+1,i+1}^m = \sum_{j=1}^{i} \left(g_{ij}^m \overline{W}_{ij}^m + v_{ij}^m s_i^m\right), \qquad 1 \le i \le n-1,\ m \in M,$$

$$\overline{H}_{i+1,i+1}^m = \sum_{j=1}^{i} \left(v_{ij}^m \overline{H}_{ij}^m + g_{ij}^m s_i^m\right), \qquad 1 \le i \le n-1,\ m \in M.$$

Следующая группа условий гарантирует, что каждый предмет размещается в области допустимого размера:

$$\begin{aligned}
&x_i \ge \overline{X}_{nj}^m - (1 - \sum_{j=1}^{n} r_{ij}^m) W_{max}, && i \in I,\ m \in M,\\
&y_i \ge \overline{Y}_{nj}^m - (1 - \sum_{j=1}^{n} r_{ij}^m) H_{max}, && i \in I,\ m \in M,\\
&w_i(1 - z_i) + h_i z_i \le \overline{W}_{nj}^m + (1 - \sum_{j=1}^{n} r_{ij}^m) W_{max}, && i \in I,\ m \in M,
\end{aligned}$$

$$h_i(1-z_i)+w_iz_i \le \overline{H}^m_{nj} + (1-\sum_{j=1}^{n} r^m_{ij})H_{max}, \qquad i \in I,\ m \in M.$$

Условия, исключающие пересечения предметов и запрещенных областей, находящихся в одном контейнере, принимают следующий вид:

$$\begin{aligned}
&x_i + w_i(1-z_i) + h_i z_i \le x^0_k + \big(1-b^L_{ik}\big)W_{max}, && i \in I,\ k \in I^0,\\
&x^0_k + w^0_k \le x_i + \big(1-b^R_{ik}\big)W_{max}, && i \in I,\ k \in I^0,\\
&y_i + h_i(1-z_i) + w_i z_i \le y^0_k + \big(1-b^B_{ik}\big)H_{max}, && i \in I,\ k \in I^0,\\
&y^0_k + h^0_k \le y_i + \big(1-b^A_{ik}\big)H_{max}, && i \in I,\ k \in I^0,\\
&b^L_{ik} + b^R_{ik} + b^B_{ik} + b^A_{ik} \ge \sum_{j=1}^{n} r^m_{ij} + q_{km} - d_i - 1, && i \in I,\ k \in I^0,\ m \in M.
\end{aligned}$$

Выписанные условия определяют неориентированную задачу с гильотинными разрезами 2BP|R|G. Аналогично при $z_i \equiv 0$ формулируется ориентированная задача 2BP|O|G. $\square$

Формулировки задач в терминах частично-целочисленного программирования позволяют использовать коммерческое программное обеспечение для нахождения оптимальных решений (см. разд. 1.3). В задаче с гильотинными разрезами некоторые ограничения имеют вторую степень, в то время как в задаче с произвольными разрезами все ограничения являются линейными. Однако даже это представляет сложность для точных методов, используемых в коммерческих пакетах (см. разд. 2.8.3). Поэтому в данной работе основные усилия направлены на разработку приближенных алгоритмов.

2.3 Кодирующие схемы

Рассматриваются две кодирующие схемы, использовавшиеся ранее в задачах проектирования интегральных микросхем [36, 71]. Содержимое каждого контейнера представляется в виде кодирующей структуры. В настоящей работе предложены алгоритмы декодирования, преобразующие такую структуру в упаковку предметов с учетом запрещенных областей.

Задачи с гильотинными разрезами возникают в приложениях, когда исходный материал необходимо разрезать при помощи гильотины. Структура

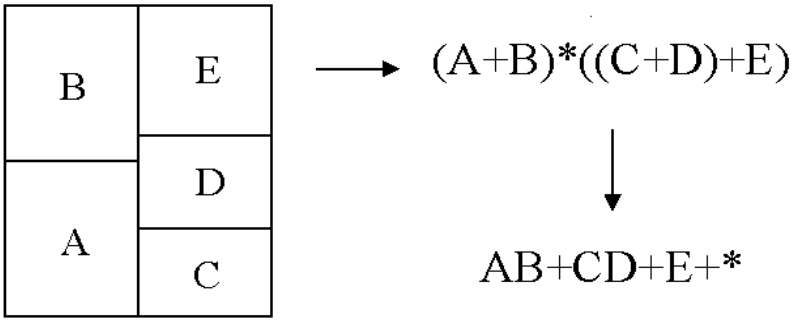


Рис. 2.3: Гильотинное решение, его структура и соответствующая ей польская запись

гильотинного решения может быть представлена в виде конечного арифметического выражения, определяющего рекурсивный процесс объединения предметов в блоки. Операндами здесь выступают прямоугольные предметы, а операторами — вертикальная, символ «+», и горизонтальная, символ «*», склейки (см. рис. 2.3). Постфиксная запись такого арифметического выражения, когда операторы стоят сразу после пары операндов, на которую они действуют, называется польской записью [71]. Любая польская запись имеет следующий вид $\alpha = \alpha_1\alpha_2 \ldots \alpha_{2n-1}$, при этом

(1) каждый символ α_i, $1 \leq i \leq 2n-1$ является либо предметом, либо оператором склейки;

(2) каждый предмет встречается в записи ровно один раз;

(3) для любого k, $1 \leq k \leq 2n-1$ количество операторов склейки в последовательности $\alpha_1 \ldots \alpha_k$ меньше количества предметов.

Решение негильотинной задачи будем называть LB–компактным, если ни один предмет в контейнере невозможно сдвинуть влево или вниз при условии, что остальные предметы остаются неподвижными [36]. Для представления LB–компактных решений будем использовать ориентированные деревья (см. рис. 2.4). Вершинам дерева соответствуют прямоугольные предметы. Ребра кодируют геометрические отношения между предметами. Между двумя вершинами ставится ребро только в том случае, когда соответствующие им предметы соприкасаются по оси x, а их проекции на ось y пересекаются. Если для некоторого предмета существует несколько потенциальных предметов-родителей, то для определенности в качестве родителя выбирается самый нижний предмет. Заметим, что среди оптимальных решений всегда найдется LB–компактное. Поэтому далее будут рассматриваться только такие решения.

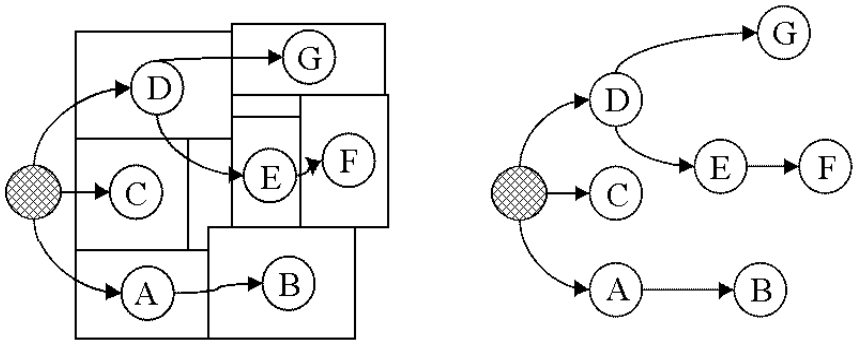


Рис. 2.4: LB-компактное решение и ориентированное дерево для него

В задачах упаковки с запрещенными областями требуется, чтобы полученное при декодировании решение было допустимым. То есть с запрещенными областями не должны пересекаться предметы, которым это не позволено по условию задачи. С этой целью разработана жадная процедура [15], позволяющая на каждом шаге алгоритма декодирования с помощью комбинаций сдвигов вверх и вправо находить наилучшее допустимое положение предмета относительно запрещенных областей. В качестве критерия выбора наилучшего положения используется функционал, зависящий от параметров предмета, его начального недопустимого и конечного допустимого положений. Например, площадь сдвига, равная площади прямоугольника, левый нижний угол которого совпадает с левым нижним углом предмета в начальном положении, а правый верхний угол совпадает с правым верхним углом предмета в конечном положении. Работа процедуры продемонстрирована на рис. 2.5. Пусть предмет B занимает недопустимое положение и пересекается с запрещенными областями (см. рис. 2.5 (а)). Тогда с помощью сдвигов вверх и вправо найдем допустимые положения $B1$, $B2$ и $B3$ (см. рис. 2.5 (б)). Положение $B2$ характеризуется минимальной площадью сдвига, поэтому разместим в нем предмет B, как в наилучшем из возможных (см. рис. 2.5 (в)). Пусть k — число запрещенных областей в контейнере. Тогда для каждого предмета требуется просмотреть не более $(k+1)^2$ положений, что обусловлено максимальным количеством сдвигов вверх равным k и таким же количеством сдвигов вправо. Таким образом трудоемкость данной процедуры для каждого контейнера составляет $O(k^2)$.

Декодирование польской записи выполняется за линейное время одно-

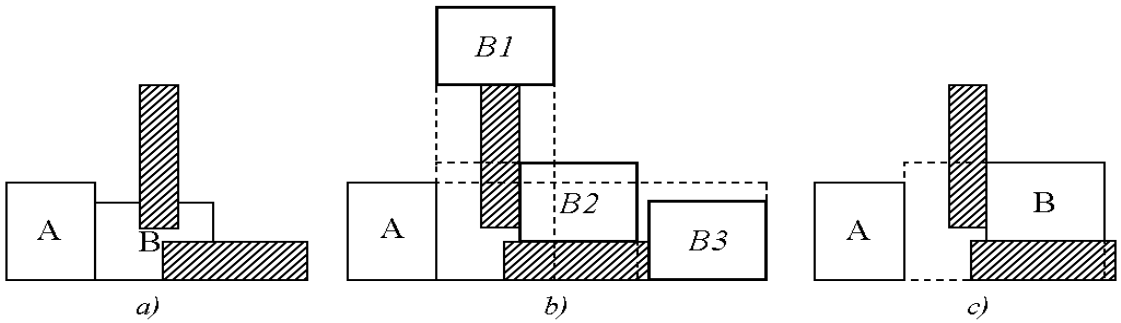


Рис. 2.5: Декодирование с учетом запрещенных областей

кратным просмотром записи и последовательной склейкой предметов соответствующими операторами, если запрещенные области отсутствуют [71]. Алгоритм декодирования выполняет следующие действия. Польская запись α просматривается слева направо. Если очередной символ записи α_i является предметом, то помещаем его в стек. Если α_i является оператором склейки, то извлекаем из стека два элемента, склеиваем их по вертикали, либо горизонтали и помещаем полученный блок в стек в качестве нового элемента. В результате, после просмотра польской записи, в стеке будет содержаться один элемент, соответствующий упаковке предметов. В задаче с запрещенными областями алгоритм декодирования становится более сложным и трудоемким. Для того, чтобы вычислить допустимое положение одного предмета необходимо полностью просмотреть польскую запись и только затем выполнить процедуру поиска допустимого положения. Чтобы получить допустимое решение, необходимо в худшем случае $\overline{n}$ раз просмотреть польскую запись и найти допустимое положение каждого предмета. Здесь $\overline{n}$ — это число предметов, содержащееся в рассматриваемом контейнере. Время работы алгоритма в этом случае становится $O(\overline{n}^2 + k^2)$.

Декодирование ориентированного дерева выполняется поиском в глубину [36]. Для каждого предмета x-координата определяется его родителем, а y-координата определяется при помощи контура частичной упаковки. Контур состоит из уже размещенных предметов, покрывающих частичную упаковку сверху. Его использование позволяет на порядок сократить время декодирования. Если предмет занимает недопустимое положение, то есть пересекается с запрещенной областью, то для него выполняется процедура поиска наилучшего допустимого положения, описанная выше, и только за-

тем продолжается декодирование. Время работы алгоритма декодирования составляет $O(\overline{n} + k^2)$.

2.4 Окрестность

В данном разделе описываются операции над польскими записями для гильотинной задачи и над ориентированными деревьями для негильотинной задачи. С помощью этих операций будут строиться окрестности решений. Выбор окрестности один из основных этапов при построении алгоритмов локального поиска. На сегодняшний день нет и, возможно никогда не будет, единого правила выбора окрестности. Кроме того для каждой задачи можно предложить несколько различных окрестностей, разных по мощности и, как следствие, с разными множествами локальных оптимумов.

Для определения операций над польскими записями понадобятся следующие понятия. Последовательность $b_1 b_2 \dots b_k$, состоящая из k операторов склейки называется цепью длины k. Цепь нулевой длины по определению является пустой последовательностью. Замыкание цепи определяется как цепь, полученная из исходной заменой операторов на противоположные («+» на «*», «*» на «+»). Пусть $\alpha = \alpha_1 \alpha_2 \dots \alpha_{2n-1}$ является польской записью. Заметим, что α может быть записана как $\pi_1 \pi_2 c_1 \pi_3 c_2 \dots c_{n-2} \pi_n c_{n-1}$, где последовательность $\pi_1 \pi_2 \dots \pi_n$ является некоторой перестановкой предметов, а c_i являются цепями суммарной длины $n-1$. Два предмета называются смежными в α, если они стоят рядом в перестановке $\pi_1 \pi_2 \dots \pi_n$. Предмет и оператор называются смежными, если они стоят рядом в последовательности $\alpha_1 \alpha_2 \dots \alpha_{2n-1}$.

При построении окрестности для гильотинной задачи будем использовать следующие операции над польскими записями:

(M1) Перестановка двух смежных предметов;

(M2) Замыкание некоторой непустой цепи операторов;

(M3) Перестановка предмета и смежного с ним оператора;

(M4) Поворот предмета на 90 градусов.

Две нормированные польские записи называются соседними, если одна получена из другой с помощью одной из указанных операций. Необходимо отметить, что некоторые операции M3 являются некорректными и могут

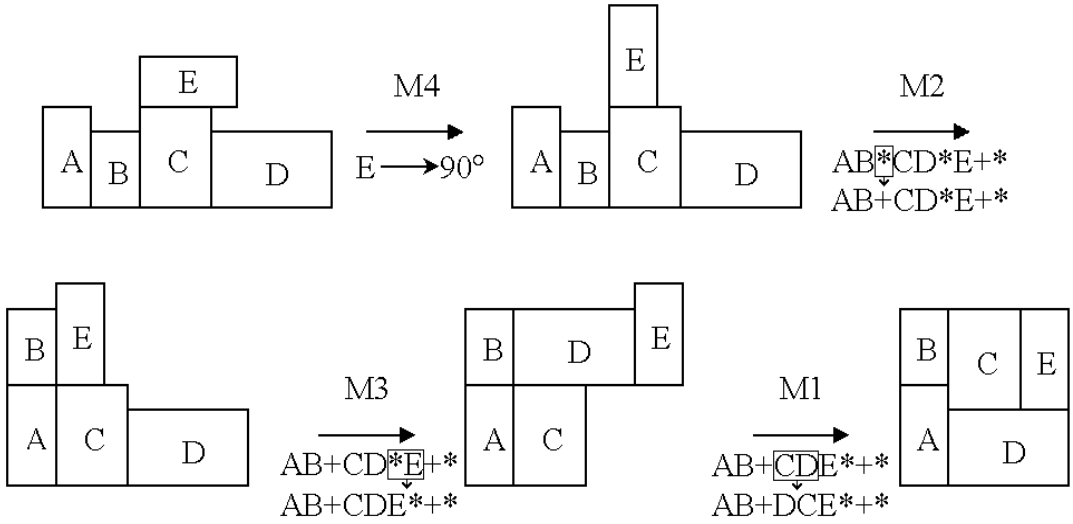


Рис. 2.6: Действие операций M1, M2, M3 и M4

привести к невыполнению условия (3) в определении польской записи. При выборе соседнего решения такие операции рассматриваться не будут. Мощность описанной окрестности является линейной.

Теорема 2.3 *Окрестность, порождаемая операциями M1, M2, M3 и M4, обладает свойством достижимости.*

Доказательство. Требуется доказать, что между двумя произвольными польскими записями α и β существует путь в окрестности M1–M4. Заметим, что операция M4, меняя ориентацию предмета, не меняет вид польской записи. Поэтому рассмотрим алгоритм, основанный на операциях M1–M3, который преобразует α к β. С помощью операций M1 и M3 можно задать произвольный порядок элементам польской записи. Операцию M2 будем использовать для преобразования цепочек операторов. Приведем с помощью M1 и M3 запись α к виду $\gamma = \pi_1\pi_2 b_1 \pi_3 b_2 \ldots b_{n-2}\pi_n b_{n-1}$, где $b_i \in \{+, *\}$. Максимальное количество шагов составляет $\frac{(n-1)(n-2)}{2}$ в случае, когда в α все операторы образуют одну цепочку, стоящую в конце записи. Применим к γ необходимое количество раз операцию M2, чтобы получить запись $\widetilde{\gamma}$ с множеством операторов, соответствующим β. Всего потребуется не более $n-1$ шагов. Затем с помощью операций M1 и M3 зададим элементам полученной записи $\widetilde{\gamma}$ порядок β, на что потребуется не более $(2n-1)(n-1)$ шагов, так как в данном случае операции M1 и M3 организуют алгоритм сортировки методом «пузырька». Таким образом потребовалось не более $\frac{(5n-2)(n-1)}{2}$ шагов для преобразования польской записи α к польской записи

β. Следовательно рассматриваемая окрестность обладает свойством достижимости. $\square$

Далее введем операции, чтобы определить окрестность для негильотинной задачи, когда решение представлено с помощью ориентированных деревьев:

(O1) Перестановка двух вершин дерева и соответствующих им предметов;

(O2) Перемещение листа дерева и соответствующего ему предмета, в новую позицию листа;

(O3) Поворот предмета на 90 градусов.

Мощность окрестности, порождаемой данными операциями является квадратичной.

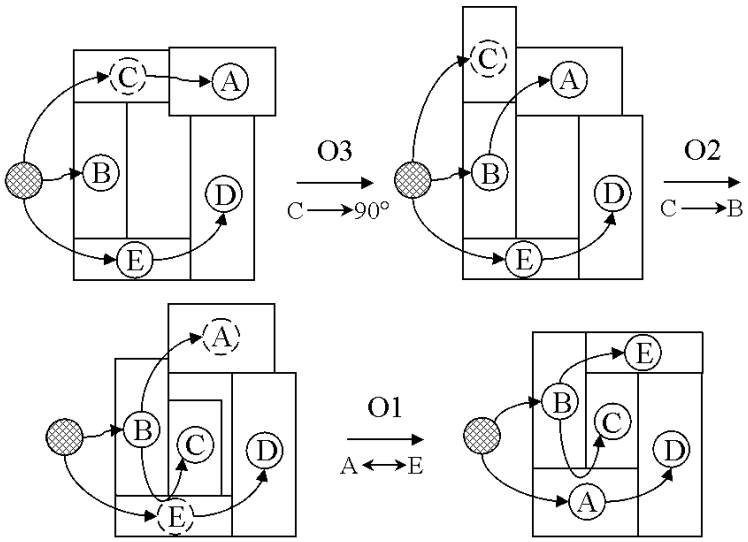


Рис. 2.7: Действие операций O1, O2 и O3

По *Теореме 1.1* окрестность O1-O3 также обладает свойством достижимости.

Действие описанных операций на упаковку предметов проиллюстрировано на рис. 2.6 и 2.7.

2.5 Модифицированный алгоритм имитации отжига

Разработанный алгоритм имитации отжига начинает свою работу с построения начального решения. Пусть число контейнеров в начальном решении

равняется m, и содержимое j-го контейнера представлено в закодированном виде S_j, $1 \leq j \leq m$. В качестве S_j может использоваться либо польская запись, либо ориентированное дерево. Далее оптимизацию можно разделить на два уровня.

Первым уровнем является параллельный локальный поиск. Для каждого контейнера цикл локального поиска повторяется заданное количество раз, равное L_j. Перед началом локального поиска данный параметр принимает значение, равное $\bar{n}$ при решении задачи с гильотинными разрезами, либо $\bar{n}^2$ при решении задачи с произвольными разрезами, где $\bar{n}$ — количество предметов в j-м контейнере. На этом уровне происходит независимое уплотнение предметов в каждом контейнере. В качестве целевой функции выступает высота упаковки контейнера $F(S_j)$. Перемещение предметов происходит только в пределах одного контейнера. Процесс поиска идет в области допустимых и недопустимых решений, то есть предметы могут выступать за границы контейнера. В этом случае в целевую функцию добавляется штраф, равный площади выступающих за границы контейнера частей предметов. Для построения окрестности используются операции, описанные в предыдущем разделе. Операция поворота предмета на 90 градусов используется при решении неориентированной задачи.

Второй уровень — выполнение алгоритма *РАЗГРУЗКА*, задача которого переместить как можно больше предметов из контейнеров стоящих в конце списка в контейнеры, находящиеся ближе к началу списка. Критерием, оценивающим качество решения при его сравнении с решением использующим такое же количество контейнеров, выступает оценка незанятой площади первых $m-1$ контейнеров, где m — это число используемых контейнеров. Данная оценка вычисляется следующим образом:

$$D_{m-1} = \frac{\sum\limits_{j=1}^{m-1}\Big(W_j H_j - \sum\limits_{i=1}^{n} w_i h_i p_{ij}\Big) - \max\Big(0, \sum\limits_{j=1}^{m-1}\sum\limits_{k=1}^{K} w_k^0 h_k^0 q_{kj} - \sum\limits_{i=1}^{n} w_i h_i d_i\Big)}{\sum\limits_{j=1}^{m-1} W_j H_j} \cdot 100\%.$$

Заметим, что в данном выражении площадь запрещенных областей, на покрытие которой не хватает предметов с параметром d_i равным единице, вычитается из незанятой площади контейнеров.

Критерием остановки алгоритма служит число итераций, либо время работы. Общая схема алгоритма представлена ниже.

Модифицированный алгоритм имитации отжига для задач прямоугольной упаковки в контейнеры

1. Построить начальное решение $S_1, \ldots, S_m$.
2. Задать начальную температуру $T > 0$.
3. Повторять, пока не выполнен критерий остановки.
   - 3.1. Выполнить цикл L_j раз для j-го контейнера, $1 \leq j \leq m$.
     - 3.1.1. Выбрать случайным образом соседнее решение S'_j из окрестности решения S_j.
     - 3.1.2. Вычислить $\Delta = F(S'_j) - F(S_j)$.
     - 3.1.3. Если $\Delta \leq 0$, то положить $S_j := S'_j$.
     - 3.1.4. Если $\Delta > 0$, то положить $S_j := S'_j$ с вероятностью $e^{-\frac{\Delta}{T}}$.
   - 3.2. Понизить температуру $T = rT$.
   - 3.3. Выполнить алгоритм *РАЗГРУЗКА*.
4. Выдать лучшее найденное решение.

2.6 Начальное решение

Начальное решение строится жадной процедурой, главный принцип которой заключается в равномерном распределении предметов среди контейнеров. На первом шаге алгоритма все контейнеры закрыты для использования. Список контейнеров пронумерован в порядке приоритета заполнения. Список предметов упорядочен по невозрастанию площади. Вычислим нижнюю оценку (см. ниже) на минимальное число контейнеров, необходимое для размещения текущего списка предметов. Откроем с начала списка число контейнеров, равное нижней оценке. Поочередно разместим предметы из списка в открытых контейнерах, пытаясь положить первый предмет сначала в первый контейнер, если не удается, то в следующий, затем второй предмет — во второй контейнер, если не удается, то в следующий и

т. д. В негильотинном случае при размещении предмета в контейнере выбирается первое допустимое положение в виде листа дерева. В гильотинном случае рассматриваются польские записи, полученные вставкой символов «$\widehat{\pi}+$», либо «$\widehat{\pi}*$» после одного из уже размещенных в контейнере предметов, где $\widehat{\pi}$ — новый предмет. Среди них выбирается первая запись, кодирующая допустимую упаковку. Если все предметы из списка удалось разместить в открытых контейнерах, то работа алгоритма завершена. Иначе шаги алгоритма повторяются, начиная с вычисления нижней оценки для неразмещенных предметов. Трудоемкость алгоритма — $O(n(nT_d + T_{LB}))$, где T_d — трудоемкость декодирования, а T_{LB} — трудоемкость вычисления нижней оценки. Качество решений, получаемых данным алгоритмом, позволяет начинать выполнение имитации отжига с низкой температурой, что существенно уменьшает время поиска. Схема алгоритма построения начального решения выглядит следующим образом.

Алгоритм построения начального решения

1. Все контейнеры закрыты для использования.
2. Упорядочить список предметов по невозрастанию их площади.
3. Найти нижнюю оценку LB оптимального числа контейнеров для задачи со списком неразмещенных предметов.
4. Открыть первые LB закрытых контейнеров.
5. Разместить предметы в открытых контейнерах в заданном списком порядке.
6. Удалить размещенные предметы из списка.
7. Если список предметов пуст, то закончить работу алгоритма, иначе перейти на шаг 3.

При решении классических задач упаковки в контейнеры в зависимости от типа решаемой задачи могут использоваться соответствующие нижние оценки для минимального числа контейнеров описанные в работах [17, 18, 26, 32]. В задачах упаковки с запрещенными областями каждый

контейнер имеет свои размеры и свое множество запрещенных областей, то есть является уникальным. Поэтому применение стандартных нижних оценок не всегда является целесообразным. Для того, чтобы учитывать уникальность контейнеров, была разработана нижняя оценка, обобщающая оценку L_0, которая также известна как непрерывная нижняя оценка для задач прямоугольной упаковки в контейнеры и вычисляется по формуле $L_0 = \left\lceil \frac{\sum_{i=1}^{N} w_i h_i}{WH} \right\rceil$, где w_i и h_i — ширина и высота i-го предмета, а W и H — ширина и высота контейнеров. Предложенная оценка указывает минимальное количество контейнеров, которое необходимо взять с начала списка, чтобы разместить все предметы.

Теорема 2.4 *Значение, определяемое величиной L_{IMP}, является нижней оценкой оптимального значения целевой функции задачи прямоугольной упаковки в контейнеры с запрещенными областями.*

$$L_{IMP} = \min\Big\{m \Big| \max\Big(0, \sum_{i=1}^{n} w_i h_i d_i - \sum_{j=1}^{m}\sum_{k=1}^{K} w_k^0 h_k^0 q_{kj}\Big) + \sum_{i=1}^{n} w_i h_i (1-d_i) - \sum_{j=1}^{m}\Big(W_j H_j - \sum_{k=1}^{K} w_k^0 h_k^0 q_{kj}\Big) \le 0,\ m \ge 0\Big\}.$$

Доказательство. При вычислении L_{IMP} используются площади предметов, контейнеров и запрещенных областей. При этом их геометрические формы не рассматриваются. Поэтому аналогично L_0 значение L_{IMP} соответствует такому количеству контейнеров, которое требуется для безотходного размещения всех предметов. В силу условий задачи такое решение невозможно улучшить. Поэтому для любого примера верно, что $L_{IMP} \le L^*$, где L^* — оптимальное количество контейнеров. Следовательно, L_{IMP} является нижней оценкой. □

2.7 Алгоритм *РАЗГРУЗКА*

Процедура разгрузки используется в предложенной схеме имитации отжига при каждом понижении температуры. Идея алгоритма заключается в том, чтобы сосредоточить мелкие предметы в контейнерах, стоящих в

конце списка, чтобы облегчить их последующую разгрузку. Алгоритм использует две операции.

Операция ЗАМЕНА(j) применяется к j-му контейнеру и заменяет содержащиеся в нем предметы на предметы большей площади из контейнеров, стоящих в конце списка, т. е. с номерами больше j, если при этом не нарушается допустимость решения.

Операция ПЕРЕМЕЩЕНИЕ(j) перемещает предметы j-го контейнера в контейнеры, стоящие в начале списка, т. е. с номерами меньше j, сохраняя допустимость решения.

Работа алгоритма заключается в последовательном применении данных операций ко всем контейнерам. Сначала к каждому контейнеру применяется операция ЗАМЕНА. В результате получаем решение, в котором контейнеры, стоящие в конце списка заполнены в основном небольшими предметами, поэтому их проще разгрузить. Операция ПЕРЕМЕЩЕНИЕ применяется к списку контейнеров, начиная с последнего. Пустые контейнеры удаляются. Время работы алгоритма составляет $O(n^2 T_d)$, где T_d — трудоемкость используемого декодера. При работе данного алгоритма предметы добавляются в контейнеры так же, как и при построении начального решения.

2.8 Численные эксперименты

Разработанный алгоритм запрограммирован на языке PASCAL и тестировался на вычислительной машине с процессором AMD Athlon 1,44 GHz как на известных тестовых примерах, так и на случайно сгенерированных.

2.8.1 Примеры прямоугольной упаковки с запрещенными областями

Для анализа работы алгоритма сгенерированы два класса примеров. В примерах первого класса (Class_Preplaced) ни один предмет не может пересекаться с запрещенными областями. Таким образом, каждая запрещенная область может рассматриваться как заранее размещенный предмет. Во втором классе (Class_Impurities) каждой запрещенной области соответствует предмет таких же размеров. При этом допускается его пересечение с за-

прещенными областями. Для каждого примера известно его оптимальное решение. Все примеры в каждом классе поделены на группы по количеству контейнеров в оптимальном решении (3, 7, 10, 15) и по количеству запрещенных областей (10, 40, либо 70 процентов от количества предметов в данном примере). Размерность примеров составляет 20, 60, 100, 140, либо 200 предметов. При генерации примеров размеры контейнеров выбирались из диапазона от 100 до 300 и затем случайным образом разрезались с помощью гильотинных разрезов на заданное количество предметов и запрещенных областей безотходным способом. Размеры дополнительных контейнеров брались равными 300×300. Указанные примеры расположены в электронной библиотеке тестовых примеров «Дискретные задачи размещения» (http://math.nsc.ru/AP/benchmarks/index.html).

Алгоритм имитации отжига начинал поиск со стартовой температурой, равной $T_0 = 10$. Коэффициент охлаждения равнялся $r = 0,99$. При формировании окрестности для каждого контейнера в зависимости от типа решаемой задачи из списка операций с равной вероятностью выбиралась одна. Критерием остановки выступало время работы. На каждый пример алгоритму отводилось 300 секунд.

В табл. 2.1 и 2.2 показаны результаты экспериментов. Решались четыре типа задач: с поворотами предметов и без них, с гильотинными и произвольными разрезами. Для всех примеров найдены либо оптимальные решения, либо решения с использованием только одного дополнительного контейнера. В таблицах указаны значения оценки D_{m-1}, характеризующей степень загрузки первых $m-1$ контейнеров. В случаях когда данная оценка достигает значения 0%, решение является оптимальным, т. к. первые $m-1$ контейнеров загружены полностью и дальнейшее улучшение упаковки невозможно. По результатам эксперимента видно, что разработанный алгоритм способен за небольшое время находить решения хорошего качества, близкие к оптимальным.

2.8.2 Примеры классической прямоугольной упаковки

Для анализа работы алгоритма на классических задачах прямоугольной упаковки в контейнеры (т. е. без запрещенных областей) использовались тестовые примеры, предложенные в работе [55].

Таблица 2.1: Значение D_{m-1} (%) для примеров с заранее размещенными предметами (Class_Preplaced)

		IMP_10				IMP_40				IMP_70			
	n	O|G	R|G	O|F	R|F	O|G	R|G	O|F	R|F	O|G	R|G	O|F	R|F
BIN_3	20	8,503	8,766	4,554	2,398	3,317	2,608	0,000	2,989	1,835	4,655	2,325	3,949
	60	8,347	6,586	5,522	4,458	8,189	6,177	7,060	5,764	7,195	5,914	7,070	5,301
	100	5,954	4,614	4,421	3,338	7,367	6,899	6,766	4,993	6,116	5,882	7,530	5,171
	140	5,577	5,337	4,571	3,986	7,762	6,407	6,156	5,355	8,850	7,812	7,729	5,538
	200	5,357	5,282	4,321	3,126	7,792	5,984	6,129	5,410	6,239	6,046	5,422	4,463
	Среднее	6,748	6,117	4,678	3,461	6,885	5,615	5,222	4,902	6,047	6,062	6,015	4,884
BIN_7	20	0,000	6,305	0,000	4,978	3,197	3,197	0,000	4,064	4,416	5,902	0,000	1,960
	60	6,656	4,135	5,558	3,147	7,409	6,163	5,878	5,008	5,876	5,441	5,309	3,966
	100	5,639	5,894	4,707	3,517	6,407	5,529	5,011	3,959	7,423	5,923	7,807	6,495
	140	6,253	5,232	4,242	4,036	7,261	6,759	6,024	5,089	7,473	6,801	6,283	4,270
	200	6,350	5,700	4,332	3,005	7,306	6,424	6,124	5,233	7,173	5,287	4,867	3,782
	Среднее	4,980	5,453	3,768	3,737	6,316	5,614	4,607	4,671	6,472	5,871	4,853	4,095
BIN_10	20	0,000	0,000	0,000	0,000	4,671	6,422	0,000	1,358	1,606	4,774	0,000	0,000
	60	6,536	4,314	4,535	3,697	7,612	5,930	5,932	4,822	5,061	4,817	4,809	4,377
	100	6,083	4,548	5,232	3,824	7,456	6,065	6,071	5,172	6,575	5,624	4,779	3,838
	140	5,418	4,289	4,800	3,959	8,231	6,874	5,903	4,894	7,256	6,127	5,543	4,726
	200	5,079	4,762	4,215	3,427	6,240	5,936	5,745	4,880	6,866	6,674	5,695	5,019
	Среднее	4,623	3,583	3,756	2,981	6,842	6,245	4,730	4,225	5,531	5,603	4,165	3,592
BIN_15	20	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	8,173	8,183	0,000	0,000
	60	6,862	4,539	4,719	3,973	5,497	5,012	4,156	3,903	7,789	5,810	4,596	4,310
	100	6,299	4,728	5,416	4,027	6,275	3,982	5,988	4,507	7,228	6,139	5,695	4,352
	140	6,038	4,214	5,377	3,991	6,327	4,830	4,883	4,471	5,104	4,922	5,190	4,475
	200	4,886	4,232	4,316	3,320	6,181	5,044	5,929	4,038	6,021	4,855	5,308	4,057
	Среднее	4,817	3,543	3,966	3,062	4,856	3,774	4,191	3,384	6,863	5,982	4,158	3,439

Таблица 2.2: Значение D_{m-1} (%) для примеров с запрещенными областями (Class_Impurities)

		IMP_10				IMP_40				IMP_70			
	n	O|G	R|G	O|F	R|F	O|G	R|G	O|F	R|F	O|G	R|G	O|F	R|F
BIN_3	20	0,000	4,550	0,000	3,224	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
	60	9,097	7,992	6,697	6,333	12,350	9,584	9,689	9,088	10,140	9,618	9,502	8,971
	100	8,300	8,576	6,119	5,899	10,637	8,720	10,110	8,348	9,940	8,684	8,434	8,414
	140	9,217	8,782	5,545	5,664	11,173	11,291	11,550	9,187	11,218	9,693	9,562	8,375
	200	8,991	9,552	7,712	7,117	12,027	11,505	11,063	9,282	10,834	8,353	8,853	7,394
	Среднее	7,121	7,890	5,215	5,647	9,237	8,220	8,482	7,181	8,426	7,270	7,270	6,631
BIN_7	20	0,000	3,206	0,000	3,206	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
	60	7,879	5,245	7,349	5,501	6,644	7,496	8,443	7,163	6,522	7,401	7,580	7,171
	100	6,425	4,400	5,507	4,701	9,982	8,292	8,994	6,831	8,300	7,356	7,678	6,557
	140	8,428	6,170	5,031	4,353	9,950	9,787	9,461	7,454	8,305	7,913	7,461	6,763
	200	8,663	7,187	6,784	3,860	10,533	9,854	9,199	7,849	7,969	6,459	6,850	5,832
	Среднее	6,279	5,242	4,934	4,324	7,422	7,086	7,219	5,859	6,219	5,826	5,914	5,265
BIN_10	20	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
	60	7,681	6,344	5,993	5,103	8,803	8,324	8,893	7,663	6,918	5,115	5,011	4,824
	100	8,269	6,566	6,088	5,002	9,917	8,682	8,814	7,061	8,451	6,642	7,242	6,386
	140	6,383	6,276	4,906	4,665	9,703	8,051	8,472	6,832	8,323	6,644	7,470	6,089
	200	8,715	7,248	5,958	4,761	9,300	7,748	7,486	5,652	8,158	6,167	7,124	6,489
	Среднее	6,210	5,287	4,589	3,906	7,545	6,561	6,733	5,442	6,370	4,914	5,369	4,758
BIN_15	20	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
	60	7,469	6,715	5,954	6,093	8,923	8,529	7,838	9,283	7,973	6,562	5,321	7,377
	100	6,452	6,560	7,395	4,875	8,291	8,233	8,057	6,276	7,032	7,779	7,646	5,941
	140	5,987	5,104	5,824	4,273	8,051	7,689	7,512	6,106	8,785	7,477	7,246	6,687
	200	7,165	5,373	5,691	4,328	9,850	9,362	9,207	7,337	8,393	7,386	8,072	6,671
	Среднее	5,415	4,750	4,973	3,914	7,023	6,763	6,523	5,800	6,437	5,841	5,657	5,335

Всего рассматривается 500 примеров, разделенных на 10 классов:

Class I: w_j и h_j равномерно распределены на отрезке [1,10], $W = H = 10$.

Class II: w_j и h_j равномерно распределены на отрезке [1,10], $W = H = 30$.

Class III: w_j и h_j равномерно распределены на отрезке [1,35], $W = H = 40$.

Class IV: w_j и h_j равномерно распределены на отрезке [1,35], $W = H = 100$.

Class V: w_j и h_j равномерно распределены на отрезке [1,100], $W = H = 100$.

Class VI: w_j и h_j равномерно распределены на отрезке [1,100], $W = H = 300$.

В описанных шести классах размеры всех предметов выбираются из одного интервала. В следующих четырех классах все предметы разделены на четыре типа:

Type 1: w_j равномерно распределено на отрезке $[\frac{2}{3}W, W]$, h_j равномерно распределено на отрезке $[1, \frac{1}{2}H]$.

Type 2: w_j равномерно распределено на отрезке $[1, \frac{1}{2}W]$, h_j равномерно распределено на отрезке $[\frac{2}{3}H, H]$.

Type 3: w_j равномерно распределено на отрезке $[\frac{1}{2}W, W]$, h_j равномерно распределено на отрезке $[\frac{1}{2}H, H]$.

Type 4: w_j равномерно распределено на отрезке $[1, \frac{1}{2}W]$, h_j равномерно распределено на отрезке $[1, \frac{1}{2}H]$.

Размеры контейнеров равны $W = H = 100$. Предметы генерируются следующим образом:

Class VII: 70% предметов первого типа и 10% предметов других типов.

Class VIII: 70% предметов второго типа и 10% предметов других типов.

Class IX: 70% предметов третьего типа и 10% предметов других типов.

Class X: 70% предметов четвертого типа и 10% предметов других типов.

Размерности примеров n во всех классах берутся равными 20, 40, 60, 80 и 100. Для каждого класса и каждого значения n рассматривается 10 примеров. Для сравнения взяты результаты, полученные с помощью алгоритма поиска с запретами [55]. При тестировании разработанного алгоритма

имитации отжига на данных примерах использовались те же начальные значения параметров, что в предыдущем эксперименте. В качестве критерия остановки также выступало время работы алгоритма, ограниченное 300 секундами.

Таблица 2.3: Среднее отношение верхних оценок к нижним для классических задач прямоугольной упаковки в контейнеры

		Поиск с запретами [55]				Имитация отжига			
Класс	n	O|G	R|G	O|F	R|F	O|G	R|G	O|F	R|F
I	20	1,11	1,03	1,06	1,05	1,00	1,03	1,00	1,03
	40	1,08	1,05	1,06	1,04	1,00	1,04	1,00	1,04
	60	1,05	1,05	1,04	1,04	1,02	1,04	1,02	1,04
	80	1,04	1,07	1,05	1,06	1,00	1,06	1,00	1,06
	100	1,05	1,04	1,04	1,03	1,00	1,02	1,00	1,02
	Среднее	1,066	1,048	1,050	1,044	1,005	1,038	1,004	1,038
II	20	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	40	1,10	1,10	1,10	1,10	1,10	1,10	1,00	1,00
	60	1,15	1,15	1,10	1,00	1,05	1,00	1,00	1,00
	80	1,07	1,03	1,07	1,03	1,03	1,03	1,00	1,00
	100	1,03	1,03	1,03	1,00	1,03	1,00	1,00	1,00
	Среднее	1,070	1,062	1,060	1,026	1,043	1,027	1,000	1,000
III	20	1,18	1,09	1,20	1,06	1,03	1,02	1,00	1,02
	40	1,12	1,11	1,11	1,09	1,06	1,09	1,03	1,07
	60	1,07	1,10	1,05	1,08	1,03	1,07	1,03	1,07
	80	1,08	1,07	1,08	1,07	1,02	1,06	1,02	1,06
	100	1,09	1,08	1,09	1,07	1,02	1,05	1,02	1,05
	Среднее	1,108	1,090	1,106	1,074	1,032	1,057	1,020	1,054
IV	20	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	40	1,10	1,00	1,00	1,00	1,10	1,00	1,00	1,00
	60	1,20	1,10	1,15	1,10	1,10	1,10	1,10	1,10
	80	1,10	1,03	1,10	1,07	1,10	1,07	1,07	1,03
	100	1,10	1,03	1,03	1,03	1,07	1,03	1,03	1,03
	Среднее	1,100	1,032	1,056	1,040	1,074	1,040	1,040	1,033
V	20	1,13	1,04	1,11	1,04	1,00	1,04	1,00	1,04
	40	1,09	1,07	1,04	1,07	1,00	1,06	1,00	1,06
	60	1,07	1,07	1,06	1,06	1,01	1,06	1,01	1,06
	80	1,08	1,08	1,06	1,07	1,03	1,07	1,03	1,07
	100	1,09	1,07	1,08	1,07	1,03	1,05	1,02	1,05
	Среднее	1,092	1,066	1,070	1,062	1,012	1,055	1,011	1,055

Таблица 2.3 — продолжение

		Поиск с запретами [55]				Имитация отжига			
Класс	n	O|G	R|G	O|F	R|F	O|G	R|G	O|F	R|F
VI	20	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	40	1,50	1,40	1,40	1,40	1,40	1,30	1,20	1,20
	60	1,10	1,05	1,05	1,05	1,05	1,05	1,00	1,00
	80	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	100	1,10	1,07	1,07	1,07	1,10	1,07	1,07	1,00
	Среднее	1,140	1,104	1,104	1,104	1,110	1,083	1,053	1,040
VII	20	1,08	1,11	1,04	1,11	1,00	1,11	1,00	1,11
	40	1,07	1,07	1,06	1,08	1,02	1,06	1,02	1,06
	60	1,05	1,06	1,05	1,06	1,01	1,04	1,01	1,04
	80	1,05	1,08	1,04	1,10	1,04	1,06	1,04	1,06
	100	1,04	1,07	1,03	1,08	1,01	1,05	1,01	1,05
	Среднее	1,058	1,078	1,044	1,086	1,016	1,063	1,016	1,063
VIII	20	1,12	1,10	1,06	1,10	1,00	1,10	1,00	1,10
	40	1,04	1,08	1,03	1,10	1,01	1,08	1,01	1,08
	60	1,03	1,07	1,02	1,07	1,03	1,05	1,01	1,04
	80	1,03	1,08	1,02	1,08	1,01	1,06	1,01	1,06
	100	1,04	1,08	1,04	1,09	1,01	1,05	1,01	1,05
	Среднее	1,052	1,082	1,034	1,088	1,010	1,067	1,007	1,065
IX	20	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	40	1,01	1,01	1,01	1,01	1,00	1,01	1,00	1,01
	60	1,01	1,01	1,01	1,01	1,00	1,01	1,00	1,01
	80	1,01	1,01	1,01	1,01	1,00	1,01	1,00	1,01
	100	1,01	1,01	1,01	1,01	1,00	1,01	1,00	1,01
	Среднее	1,008	1,008	1,008	1,008	1,000	1,008	1,000	1,008
X	20	1,14	1,12	1,10	1,12	1,02	1,13	1,00	1,12
	40	1,09	1,08	1,06	1,06	1,00	1,06	1,00	1,06
	60	1,08	1,07	1,07	1,06	1,05	1,06	1,04	1,06
	80	1,10	1,06	1,06	1,05	1,06	1,05	1,06	1,04
	100	1,07	1,05	1,08	1,05	1,05	1,04	1,04	1,03
	Среднее	1,096	1,076	1,074	1,068	1,035	1,067	1,026	1,064

В табл. 2.3 представлены средние значения отношений полученных верхних оценок к нижним, а также результаты, опубликованные в работе [55]. Для 69,7% примеров верхние оценки совпали с нижними, что означает, что для данных примеров были найдены оптимальные решения. Также с помощью разработанного алгоритма были улучшены результаты, полученные ранее.

В табл. 2.4 указано среднее время нахождения рекордного решения для примеров из десяти рассматриваемых классов. Алгоритм поиска с запретами выполнялся на вычислительной машине с процессором Silicon Graphics

Таблица 2.4: Среднее время (сек.) нахождения последнего рекордного решения алгоритмами для классических задач прямоугольной упаковки в контейнеры

	Поиск с запретами [55]				Имитация отжига			
Класс	O|G	R|G	O|F	R|F	O|G	R|G	O|F	R|F
I	50,50	39,46	43,60	35,60	3,44	3,85	2,51	2,35
II	3,61	2,46	3,62	1,22	0,68	0,46	0,58	0,42
III	54,96	50,49	53,62	46,91	4,86	4,48	4,27	4,04
IV	7,23	2,48	4,83	3,62	0,97	0,45	0,73	0,52
V	53,70	43,52	47,28	42,70	5,85	5,22	4,16	4,92
VI	3,61	2,43	2,41	2,42	0,56	0,39	0,47	0,41
VII	42,38	50,04	37,48	51,60	2,71	2,05	1,67	1,78
VIII	38,28	50,08	29,67	52,13	2,17	2,84	1,86	1,52
IX	25,48	19,51	27,35	20,86	0,81	0,75	0,74	0,73
X	45,65	32,31	37,34	28,58	3,16	3,33	2,45	2,10

INDY R10000sc. Время его работы было ограничено 60 секундами. Время работы алгоритма имитации отжига было ограничено 300 секундами на вычислительной машине, указанной в начале раздела. Однако алгоритм имитации отжига использует целевую функцию D_{m-1}, характеризующую плотность заполнения предметами первых $m-1$ контейнеров, т. к. она более чувствительна к локальным изменениям решения. Поэтому в табл. 2.4 указано время последнего нахождения решения с меньшим количеством контейнеров. Используя таблицу быстродействия ЭВМ [29], несложно заметить, что время нахождения последнего рекордного решения для алгоритма [55] на вычислительной машине с процессором AMD Athlon 1,44 GHz имело бы тот же порядок величины, что и время предложенного алгоритма.

2.8.3 Использование пакета GAMS

Данный раздел содержит результаты, полученные с помощью коммерческого пакета GAMS. Расчеты велись на вычислительной машине с процессором Intel Celeron 1,8 GHz. Рассмотрены 500 тестовых примеров классической прямоугольной упаковки из предыдущего раздела. На поиск решения пакету отводилось 20 часов для каждого примера. В качестве решающего ядра для модели с произвольными разрезами выбран CPLEX (версия 11.0).

Таблица 2.5: Результаты, полученные пакетом GAMS

Класс	n	GAMS$_{\frac{UB}{LB}}$	ОПТ$_{GAMS}$	ОПТ$_{UB=LB}$	Класс	n	GAMS$_{\frac{UB}{LB}}$	ОПТ$_{GAMS}$	ОПТ$_{UB=LB}$
I	20	1,00	5	10	VI	20	1,00	10	10
	40	1,10	—	3		40	1,40	1	8
	60	1,30	—	—		60	7,95	—	—
	80	1,30	—	—		80	9,42 (4)	—	—
	100	1,30 (3)	—	—		100	20,00 (1)	—	—
II	20	1,00	10	10	VII	20	1,00	1	10
	40	1,20	—	7		40	1,26	—	—
	60	6,53	—	—		60	1,71	—	—
	80	12,08 (4)	—	—		80	1,78	—	—
	100	14,75 (4)	—	—		100	1,77 (1)	—	—
III	20	1,00	8	10	VIII	20	1,00	1	10
	40	1,15	—	3		40	1,36	—	—
	60	1,70	—	—		60	1,90	—	—
	80	1,72	—	—		80	1,79	—	—
	100	— (0)	—	—		100	1,86 (4)	—	—
IV	20	1,00	10	10	IX	20	1,00	10	10
	40	1,20	—	7		40	1,00	8	10
	60	6,52	—	—		60	1,00	9	10
	80	11,87 (5)	—	—		80	1,00	6	10
	100	17,94 (3)	—	—		100	1,00 (7)	7	7
V	20	1,00	10	10	X	20	1,08	6	8
	40	1,10	—	5		40	1,19	—	1
	60	1,34	—	—		60	2,37	—	—
	80	1,33 (6)	—	—		80	2,61	—	—
	100	1,51 (7)	—	—		100	3,07 (1)	—	—

В силу нелинейности ограничений для модели с гильотинными разрезами выбран BARON (версия 8.1.1). Для примеров с количеством предметов больше 40 решатель CPLEX использовался в режиме поиска приближенных решений (значение параметра *mipemphasis=1*). Рассматривались постановки без поворотов предметов. В результате расчетов оказалось, что модель с гильотинными разрезами является слишком сложной для пакета, т.к. не было найдено ни одной гильотинной упаковки для рассматриваемых примеров. В табл. 2.5 показаны результаты, полученные для задачи с произвольными разрезами. Приводятся средние значения отношений полученных верхних оценок к известным нижним. Случаи, когда не для

всех примеров были найдены допустимые решения, отмечены числами в скобках, указывающими сколько примеров данной размерности в классе было решено. В следующих двух столбцах указывается сколько примеров из десяти в каждом подклассе было решено оптимально. При этом в первом столбце ОПТ_{GAMS} указывается количество примеров, для которых пакету удалось доказать оптимальность найденного решения, а во втором $\text{ОПТ}_{UB=LB}$ количество примеров, для которых найденная пакетом верхняя оценка совпала с известной нижней оценкой.

В результате вычислительного эксперимента решены 82% рассмотренных тестовых примеров. Для 20,4% примеров пакету удалось доказать, что найденное решение является оптимальным. Оптимальность еще 13,4% установлена совпадением найденного решения с известной нижней оценкой. Как видно из таблицы, примеры размерностью 20 предметов достаточно эффективно решаются с помощью пакета. Сложность примеров значительно возрастает с увеличением числа предметов. Начиная с размерности 80 предметов, не всегда удается получить допустимое решение, не говоря уже о качественном приближенном решении.

2.9 Выводы к главе 2

Рассмотрена новая прикладная задача прямоугольной упаковки в контейнеры с запрещенными областями. Исследованы формулировки с гильотинными и произвольными разрезами. Выписаны математические постановки данных задач в терминах частично-целочисленного линейного и квадратичного программирования, что позволяет исследовать коммерческое программное обеспечение. Разработан алгоритм имитации отжига, использующий кодирующие схемы *Польская запись* и *Ориентированное дерево*. Для каждой кодировки предложен оригинальный декодер, восстанавливающий упаковку предметов с учетом запрещенных областей. Разработан жадный алгоритм построения начального решения, позволяющий начинать поиск с низкой температуры. Алгоритм имитации отжига использует окрестности линейной и квадратичной мощности. При каждой смене температуры выполняется процедура уплотнения, позволяющая концентрировать небольшие предметы в отдельных контейнерах, что облегчает их последующую

разгрузку. Проведенные численные эксперименты показали, что алгоритм позволяет находить решения с малой погрешностью, в том числе и глобально оптимальные, что свидетельствует о его эффективности.

Глава 3

Алгоритм вероятностного поиска с запретами для задачи упаковки кругов и прямоугольников в полосу

3.1 Введение

В последней главе рассматривается задача упаковки конечного множества кругов и прямоугольников в полубесконечную полосу заданной ширины. Каждый круг задается радиусом, а каждый прямоугольник задается длиной и шириной. Требуется разместить в полосе все предметы без пересечений так, чтобы длина занятой части полосы была минимальной. Стороны прямоугольников при размещении должны быть параллельны сторонам полосы. Также допускаются повороты прямоугольников на 90 градусов. Сформулированная задача является обобщением известной $\mathbb{NP}$–трудной задачи одномерной упаковки в контейнеры [5] и также является $\mathbb{NP}$–трудной в сильном смысле.

Рассматриваемая задача относится к классу задач двумерного раскроя и упаковки. Известны различные постановки задач из этого класса [8, 9, 30, 54]. Для упаковки только прямоугольников, либо только кругов получено немало результатов. Однако задача упаковки одновременно кругов и прямоугольников остается малоизученной. В работе [37] рассматривается задача упаковки в полосу кругов, прямоугольников и более сложных фигур различных размеров. Для ее решения используется генетический алгоритм, основанный на эвристической процедуре преобразования перестановки предметов в упаковку. В [66] предлагается алгоритм

нахождения глобального оптимума задачи упаковки в полосу кругов и прямоугольников и ее обобщений. Предложенный подход является комбинацией метода ветвей и границ и метода градиентов.

Наиболее известные постановки задач двумерной прямоугольной упаковки связаны с упаковкой различных прямоугольников в полосу минимальной длины, либо в минимальное число одинаковых прямоугольных контейнеров. Наиболее изученным является ориентированный случай задачи, когда повороты прямоугольников запрещены, а их стороны параллельны сторонам контейнеров или полосы. Подробный обзор, посвященный математическим моделям, нижним оценкам, приближенным и точным алгоритмам, а также реализациям метаэвристик для ориентированного случая задачи прямоугольной упаковки в полосу, либо в контейнеры можно найти в [54]. Другие варианты задачи, когда допускаются повороты прямоугольников на 90 градусов, а также ограничения на размещение прямоугольников, такие как «гильотинный раскрой», подробно освещаются в [55, 56]. Из последних результатов особый интерес представляют следующие работы. В [16] рассматривается задача упаковки прямоугольников в выпуклую область. Данная задача формулируется в работе в терминах нелинейного программирования. Для ее решения авторы используют численные методы. В [19] предлагается генетический алгоритм для решения как ориентированного, так и неориентированного случаев задачи прямоугольной упаковки в полосу. Для этой же задачи в [13] разработан гиперэвристический алгоритм, объединяющий в себе принципы нескольких эвристик. В [75] для неориентированной прямоугольной упаковки в полосу описывается генетический алгоритм, использующий разбиение исходной задачи на подзадачи меньшей размерности. Для ориентированного случая задачи прямоугольной упаковки в полосу в [12] предлагается метод GRASP, а в [44] — алгоритм локального поиска, использующий процедуру вычисления целевой функции, основанную на динамическом программировании. В [41] для решения задачи прямоугольной упаковки в контейнер заданного размера предлагается жадный полиномиальный алгоритм, который авторам также удалось эффективно применить к задаче прямоугольной упаковки в полосу. Что касается точных методов, то в [50] для ориентированного и неориентированного случаев задачи прямоугольной упаковки

в полосу предлагается метод ветвей и границ. Предложенный метод показал наибольшую эффективность на примерах безотходной упаковки прямоугольников, а также в некоторых случаях позволил найти оптимальные решения для общей задачи.

Среди известных постановок задач двумерной упаковки кругов можно выделить следующие. В [67] рассматривается задача упаковки кругов различных размеров в полосу минимальной длины и предлагается алгоритм для ее точного решения. Существует ряд статей, посвященных задаче упаковки различных кругов в двумерный рюкзак, в качестве которого выступает прямоугольник, либо другая геометрическая фигура заданного размера. Цель задачи — так заполнить рюкзак заданными кругами, чтобы минимизировать неиспользуемое в рюкзаке место. Например, в [34] рассматривается задача упаковки труб разного радиуса в прямоугольный контейнер, что эквивалентно упаковке различных кругов в прямоугольный рюкзак. Для ее решения предложены несколько полиномиальных эвристик, а также генетический алгоритм, использующий кодирование решений с помощью перестановок возможных положений кругов относительно друг друга. В этой же работе приводится формулировка задачи в терминах целочисленного нелинейного программирования. В [42, 43] авторы предлагают жадные полиномиальные алгоритмы упаковки различных кругов в заданный контейнер прямоугольной или круглой формы. В [38] авторы рассматривают задачу упаковки в прямоугольный рюкзак круглых деталей нескольких типов с учетом максимального спроса на них. Для нахождения приближенного решения предлагается генетический алгоритм.

Также существуют постановки, где требуется минимизировать площадь, в которой размещаются предметы. Например, в [57] рассматривается задача об упаковке одинаковых кругов в прямоугольник минимальной площади. Для ее решения в работе приводятся два алгоритма: один имитирует физическое взаимодействие кругов под действием силы, направленной на сжатие прямоугольника, другой основан на переборе возможных шаблонов упаковки кругов. В [74] предлагается интересный подход к задаче упаковки кругов разных размеров в минимальный квадрат. Сформулировав задачу в терминах нелинейного программирования и установив условия оптимальности первого порядка, авторы применяют модифицированный

метод Лагранжа для ее решения. В работах [1, 11] для получения новых композиционных материалов с заранее заданными свойствами используется более общая модель упаковки твердых шаров. Из последних работ стоит отметить [48], в которой приводится математическая постановка задачи упаковки кругов и выпуклых многоугольников в несколько прямоугольников с минимальной суммарной площадью. При этом допускаются повороты многоугольников на произвольный угол. Для решения поставленной задачи используется коммерческое программное обеспечение.

В данной главе предлагается вероятностный алгоритм поиска с запретами для задачи упаковки заданного множества кругов и прямоугольников в полосу минимальной длины. Алгоритм использует так называемые двухконтактные кодировки, сокращающие пространство решений, тем самым, уменьшая время поиска приближенного решения задачи. Настоящая глава организована следующим образом. В разд. 3.2 приводится математическая постановка задачи. В разд. 3.3 вводится понятие двухконтактной схемы кодирования, рассматриваются некоторые ее свойства. В разд. 3.4 определяется окрестность решений. В разд. 3.5 описывается разработанный алгоритм вероятностного поиска с запретами. В разд. 3.6 приводятся результаты численных экспериментов.

3.2 Математическая постановка задачи

Обозначим исходные данные задачи следующими параметрами: W — ширина полосы; n_c — число кругов; n_r — число прямоугольников; $I = \{1, 2, \ldots, n_c\}$ — множество кругов; $J = \{1, 2, \ldots, n_r\}$ — множество прямоугольников; $r_i,\ i \in I$, — радиус i-го круга; l_j и $w_j,\ j \in J$ — длина и ширина j-го прямоугольника.

Введем переменные задачи. Пусть x_i^c и y_i^c — координаты центра i-го круга, а x_j^r и y_j^r — координаты левого нижнего угла j-го прямоугольника. Для учета поворотов прямоугольников на 90 градусов введем бинарную переменную z_j, принимающую значение 1, если j-й прямоугольник повернут на 90 градусов, и значение 0 в противном случае. Пусть бинарная переменная a_{jm}^L равняется единице, если j-й прямоугольник находится левее m-го, и нулю в противном случае. Аналогично бинарная переменная a_{jm}^B

равняется единице, если j-й прямоугольник находится ниже m-го, и нулю в противном случае. В качестве целевой функции будет выступать длина занятой части полосы L. Тогда математическая постановка задачи будет записана следующим образом:

найти $\min L$ при ограничениях:

$$\begin{aligned} &x_i^c + r_i \le L, && i \in I, \\ &x_j^r + l_j(1 - z_j) + w_j z_j \le L, && j \in J. \end{aligned}$$

Выпишем условия, гарантирующие, что все круги будут размещены в пределах полосы:

$$\begin{aligned} &x_i^c - r_i \ge 0, && i \in I, \\ &y_i^c - r_i \ge 0, && i \in I, \\ &y_i^c + r_i \le W, && i \in I. \end{aligned}$$

По аналогии выпишем соответствующие условия для прямоугольников:

$$\begin{aligned} &x_j^r \ge 0, && j \in J, \\ &y_j^r \ge 0, && j \in J, \\ &y_j^r + w_j(1 - z_j) + l_j z_j \le W, && j \in J. \end{aligned}$$

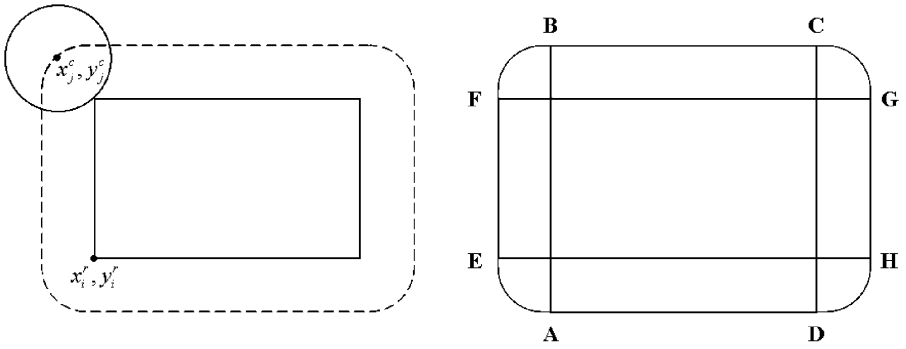
Следующее условие исключает пересечения кругов между собой. Два круга не пересекутся только тогда, когда расстояние между их центрами будет не меньше суммы их радиусов:

$$\left(x_i^c - x_k^c\right)^2 + \left(y_i^c - y_k^c\right)^2 \ge (r_i + r_k)^2, \ i, k \in I.$$

Следующие три условия исключают пересечения прямоугольников друг с другом. В первом неравенстве используется L_{max} — достаточно большое число, $L_{max} > L$:

$$\begin{aligned} &x_j^r + l_j(1 - z_j) + w_j z_j \le x_m^r + (1 - a_{jm}^L) L_{max}, && j, m \in J, \\ &y_j^r + w_j(1 - z_j) + l_j z_j \le y_m^r + (1 - a_{jm}^B) W, && j, m \in J, \\ &a_{jm}^L + a_{mj}^L + a_{jm}^B + a_{mj}^B = 1, && j, m \in J. \end{aligned}$$

Остальные условия необходимы, чтобы исключить пересечения кругов и прямоугольников. Круг i пересекается с прямоугольником j тогда и только тогда, когда центр круга лежит внутри области, граница которой обозначена пунктиром на рис. 3.1 (а). Если центр круга лежит на границе, то



(а) (б)

Рис. 3.1: Условие непересечения прямоугольника и круга

фигуры соприкасаются. Следовательно, необходимо выписать условия, которые исключают нахождение точки x_i^c, y_i^c внутри этой области. Заметим, что интересующую нас область можно покрыть более простыми фигурами, а именно двумя пересекающимися прямоугольниками ABCD, EFGH и четырьмя кругами радиуса r_i с центрами в вершинах прямоугольника j (см. рис. 3.1 (б)). Таким образом, точка x_i^c, y_i^c не должна лежать внутри ни одного из четырех кругов, ни внутри прямоугольника ABCD, ни внутри прямоугольника EFGH.

Следующая группа неравенств гарантирует, что центр i-го круга лежит за пределами двух покрывающих прямоугольных областей j-го прямоугольника. Дизъюнкции условий, реализуется с помощью бинарных переменных b_{ij}^t и c_{ij}^t:

$$
\begin{aligned}
& x_i^c + r_i \le x_j^r + (1 - b_{ij}^1) L_{max}, && i \in I,\ j \in J,\\
& x_j^r + l_j(1 - z_j) + w_j z_j \le x_i^c - r_i + (1 - b_{ij}^2) L_{max}, && i \in I,\ j \in J,\\
& y_i^c \le y_j^r + (1 - b_{ij}^3) W, && i \in I,\ j \in J,\\
& y_j^r + w_j(1 - z_j) + l_j z_j \le y_i^c + (1 - b_{ij}^4) W, && i \in I,\ j \in J,\\
& \sum_{t=1}^{4} b_{ij}^t = 1, && i \in I,\ j \in J,\\
& x_i^c \le x_j^r + (1 - c_{ij}^1) L_{max}, && i \in I,\ j \in J,\\
& x_j^r + l_j(1 - z_j) + w_j z_j \le x_i^c + (1 - c_{ij}^2) L_{max}, && i \in I,\ j \in J,\\
& y_i^c + r_i \le y_j^r + (1 - c_{ij}^3) W, && i \in I,\ j \in J,\\
& y_j^r + w_j(1 - z_j) + l_j z_j \le y_i^c - r_i + (1 - c_{ij}^4) W, && i \in I,\ j \in J,
\end{aligned}
$$

$$\sum_{t=1}^{4} c_{ij}^t = 1, \qquad\qquad i \in I,\ j \in J.$$

Последние четыре условия необходимы, чтобы расстояние между центром круга i и вершинами прямоугольника j было не меньше r_i:

$$\begin{aligned}
&\left(x_i^c - x_j^r\right)^2 + \left(y_i^c - y_j^r\right)^2 \geq r_i^2, && i \in I,\ j \in J,\\
&\left(x_i^c - \left(x_j^r + l_j(1 - z_j) + w_j z_j\right)\right)^2 + \left(y_i^c - y_j^r\right)^2 \geq r_i^2, && i \in I,\ j \in J,\\
&\left(x_i^c - x_j^r\right)^2 + \left(y_i^c - \left(y_j^r + w_j(1 - z_j) + l_j z_j\right)\right)^2 \geq r_i^2, && i \in I,\ j \in J,\\
&\left(x_i^c - \left(x_j^r + l_j(1 - z_j) + w_j z_j\right)\right)^2 + \left(y_i^c - \left(y_j^r + \right.\right. && \\
&\qquad\qquad \left.\left. + w_j(1 - z_j) + l_j z_j\right)\right)^2 \geq r_i^2, && i \in I,\ j \in J.
\end{aligned}$$

Поставленная задача относится к классу частично-целочисленных квадратичных проблем и может быть решена с помощью пакетов, использующих методы глобальной и локальной оптимизации. Однако существующие на сегодняшний день пакеты справляются лишь с примерами небольшой размерности (подробнее об этом в разд. 5.3). Поэтому нашей целью является разработка приближенного алгоритма, способного для примеров достаточно большой размерности за небольшое время находить решения, близкие к оптимальным. Алгоритмы, называемые *метаэвристиками*, относятся к одному из эффективных с этой точки зрения классов алгоритмов [31, 35]. Метаэвристики могут применяться ко многим задачам комбинаторной оптимизации, так как описаны в достаточно общих терминах. Для реализации многих метаэвристик на практике требуется определить такие ключевые термины, как представление решения и его окрестность. Как правило, решение задачи представляется в виде некоторой удобной структуры данных, по которой его можно однозначно восстановить. По окрестности определяются соседние решения для текущего решения. От нее также зависит трудоемкость каждого шага алгоритма. Следующие два раздела посвящены определению этих понятий для рассматриваемой задачи.

3.3 Двухконтактные кодировки

Как уже не раз было отмечено, важную роль в задачах упаковки играет способ представления (кодирования) допустимых решений. Самая есте-

ственная кодировка, которая указывает координаты центров фигур, уступает многим другим по ряду показателей, например, допускает наложение предметов. Кроме того, существует бесконечное число всевозможных решений, представленных таким способом. Поэтому имеет смысл отказаться от ее использования и кодировать решения с помощью более сложных структур данных. Для задач двумерной прямоугольной упаковки разработано множество различных кодировок, отличающихся как трудоемкостью декодирования, так и мощностью множества рассматриваемых решений. Для задач прямоугольной упаковки в полосу решения зачастую представляются в виде перестановок предметов. Каждая перестановка преобразуется в упаковку предметов с помощью полиномиального алгоритма декодирования. В [40] рассматриваются два таких алгоритма: *Нижний-Левый* с трудоемкостью $O(n^2)$ и *Нижний-Левый-Модифицированный* с трудоемкостью $O(n^3)$, где n — число предметов. Также в [54, 55, 56] можно найти описания различных способов преобразования перестановки предметов в упаковку для задач прямоугольной упаковки в контейнеры. В задачах прямоугольной упаковки, возникающих при проектировании интегральных микросхем, в качестве кодировок используются более сложные структуры данных и соответствующие им алгоритмы декодирования. Посвященный этой теме обзор содержится в [53]. Для задачи упаковки кругов в [34] предложено рассматривать множество двухконтактных решений, являющееся подмножеством исходного. Круги упаковываются в прямоугольный контейнер в заданном порядке так, чтобы каждый следующий круг имел как минимум две точки касания с уже упакованными кругами, либо границами контейнера. В [37, 38] похожая идея используется для задач упаковки кругов, прямоугольников и предметов более сложной формы. Порядок предметов задается перестановкой, а положение каждого следующего предмета при упаковке определяется парой уже упакованных предметов. В процессе декодирования перестановки предметов в упаковку используются сдвиги предметов, параллельные осям координат, что существенно увеличивает трудоемкость декодирования. В данной работе предложена схема кодировки двухконтактных решений, основанная на перестановке предметов и использующая более простой и менее трудоемкий алгоритм декодирования чем в [37, 38]. Упаковка предметов при заданной перестановке осуществ-

ляется следующим образом.

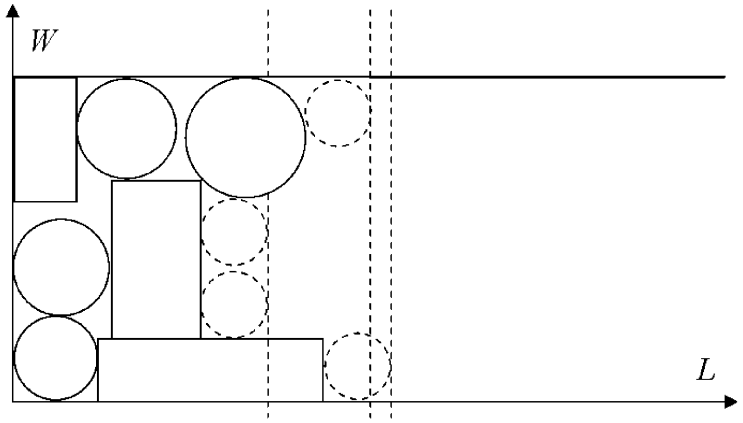


Рис. 3.2: Построение двухконтактного решения

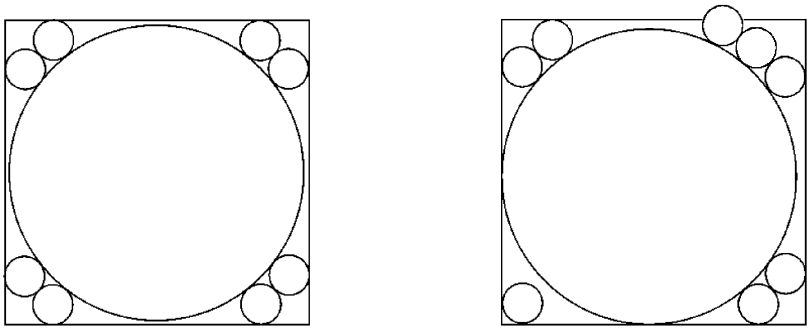
Первый предмет помещается в левый нижний угол полосы. Каждый следующий предмет размещается так, чтобы он имел как минимум две точки касания с уже упакованными предметами, либо границами полосы. Пересечения предметов и их выход за границы полосы не допускаются. Если для очередного предмета таких положений несколько, то выбирается положение с минимальным увеличением длины полосы (рис. 3.2). Прямоугольники при поиске допустимых положений рассматриваются как в их начальной ориентации, так и с поворотом на 90 градусов. Таким образом, ориентация прямоугольников определяется в процессе декодирования. В случае, когда минимальному увеличению длины полосы соответствует несколько положений, для определенности алгоритм декодирования выбирает нижнее положение. Трудоемкость декодирования оценивается величиной $O(n^4)$, где n — число всех предметов. Мощность описанной кодировки, т. е. количество всех кодов, равна мощности множества всевозможных перестановок предметов, что составляет $n!$.

Следующая теорема отвечает на вопрос: "Всегда ли множество оптимальных решений задачи содержит упаковку двухконтакнтого типа?"

Теорема 3.1 *Существует семейство входных данных задачи упаковки кругов в полосу минимальной длины, на которых множество двухконтактных решений не содержит оптимум.*

Доказательство. Рассмотрим пример упаковки девяти кругов в полосу ширины $2(R+\varepsilon)$. Радиус одного круга равен R. Радиусы оставшихся восьми кругов равны $r = (5-\sqrt{2})R+(2-\sqrt{2})\varepsilon-\sqrt{2(10-7\sqrt{2})(2R+\varepsilon)R}$.

Параметры R и ε выбраны так, чтобы выполнялось условие $R + 2\varepsilon < 3r + 2\sqrt{(R+\varepsilon)(r-\varepsilon)}$. На рис. 3.3 (а) показана упаковка всех кругов в квадрат со стороной $2(R+\varepsilon)$. Такое решение не является двухконтактным, так как в левом нижнем углу нет круга, касающегося двух сторон квадрата. Оптимальное среди двухконтактных решение показано на рис. 3.3 (б). Данная упаковка найдена перебором всех вариантов. Несложно заметить, что требуется увеличение длины полосы для того, чтобы разместить последний круг. Следовательно, на данном классе примеров множество двухконтактных решений не содержит оптимум. □



(а) (б)

Рис. 3.3: Не двухконтактное (а) и оптимальное среди двухконтактных (б) решения

Таким образом, поиск в области двухконтактных упаковок не гарантирует нахождения оптимального решения поставленной задачи. Тем не менее, эта кодировка дает экспоненциальное число кодов. И дальнейшие усилия будут направлены на поиск наилучшего решения в этом множестве.

В [49] исследовался вопрос о сравнении описанной двухконтактной кодировки для упаковки кругов в полосу с другой более мощной двухконтактной кодировкой из [34], мощность которой составляла $O\big(\frac{n!(n+1)!(n+2)!}{2^{n+1}}\big)$. При малой размерности задач более детальная кодировка имеет небольшие преимущества по точности получаемых решений, но уже при $n > 30$ это преимущество теряется и кодировка с меньшей мощностью оказывается предпочтительнее для методов локального поиска. В связи с этим далее будет использоваться двухконтактная кодировка мощностью $n!$. Вопрос о представлении решений рассматриваемой задачи упаковки с помощью кодировки, обладающей конечным пространством кодов и гарантирующей наличие кода для оптимального решения задачи, пока остается открытым.

3.4 Окрестность

Введем операции над перестановками, с помощью которых строится окрестность решений. Выбор окрестности играет важную роль при построении алгоритмов локального поиска. От него существенно зависит трудоемкость одного шага алгоритма, общее число шагов и, в конечном счете, погрешность получаемого решения.

Будем использовать следующие операции над перестановкой предметов.

СДВИГ(i, j) ставит предмет i в позицию предмета j, при этом все находящиеся между ними предметы, включая j, перемещаются на одну ячейку в сторону первоначальной позиции предмета i.

ЗАМЕНА(i, j) меняет местами в перестановке предметы i и j.

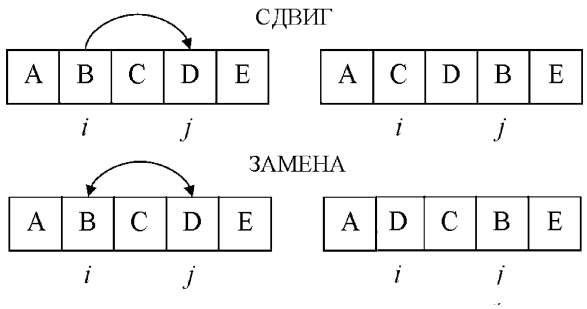


Рис. 3.4: Действие операций СДВИГ и ЗАМЕНА

Две перестановки называются *соседними*, если одна получена из другой при помощи одной из этих двух операций. *Окрестностью* текущего решения будем называть множество решений, полученное в результате применения к нему данных операций для всевозможных пар (i, j). Как показывают численные эксперименты, использование только одной операции дает худшие результаты, чем использование обеих. Операции оказывают разное влияние на упаковку предметов, соответствующую некоторой перестановке, и в совокупности порождают достаточно эффективную окрестность мощности $O(n^2)$.

3.5 Алгоритм вероятностного поиска с запретами

Алгоритм, разработанный для решения задачи упаковки кругов и прямоугольников в полосу, основан на принципах поиска с запретами, осно-

воположником которого является Ф. Гловер [35]. Алгоритм поиска с запретами представляет собой вероятностную процедуру локального поиска и принадлежит к классу метаэвристик. Для работы алгоритмов локального поиска требуется определить окрестность решения. В нашем случае окрестностью решения i_k будем называть множество решений $N(i_k)$, полученных путем однократного применения операции либо СДВИГ, либо ЗАМЕНА к данному решению i_k. Рассмотрим также рандомизированную окрестность $N_P(i_k) \subseteq N(i_k)$, где каждый элемент окрестности $N(i_k)$ включается в множество $N_P(i_k)$ с вероятностью $0 \leq P \leq 1$ независимо от других элементов. С ненулевой вероятностью множество $N_P(i_k)$ может совпадать с $N(i_k)$, может оказаться пустым или содержать ровно один элемент. Алгоритм поиска с запретами, представленный в данном разделе, осуществляет вероятностный локальный поиск по рандомизированной окрестности, совершая шаги как улучшающие целевую функцию, так и ухудшающие ее, что позволяет алгоритму не останавливаться в точке локального оптимума, как это предписано в алгоритме локального спуска, а перемещаться от одного локального оптимума к другому с целью найти среди них лучшее решение. Основным механизмом, позволяющим алгоритму покидать локальные оптимумы, является список запретов $\mathrm{TABU}_l(i_k)$. Он строится по предыстории поиска, т. е. по нескольким последним итерациям, и запрещает часть окрестности $N(i_k)$ текущего решения. На каждом шаге алгоритма очередная точка i_{k+1} является оптимальным решением подзадачи $f(i_{k+1}) = \min\{f(j) \mid j \in N_P(i_k) \backslash \mathrm{TABU}_l(i_k)\}$, где $f(i)$ — функционал, выступающий критерием качества решения. Окрестность текущего решения ограничивается теми решениями, которые не запрещены списком $\mathrm{TABU}_l(i_k)$. На каждой итерации среди них выбирается лучшее решение в качестве нового текущего решения. В стандартном алгоритме поиска с запретами это решение добавляется в список запретов, при этом из него удаляется одно из ранее добавленных решений. В разработанном алгоритме список запретов формируется из тех фрагментов решения, которые менялись на последних l шагах алгоритма, тем самым запрещая их использование. Другими словами, каждый элемент списка запретов представляет собой множество соседних решений, заданных парой операций, позволяющих перейти от текущего решения к соседнему и обратно. В случае, когда

новое решение получено с помощью операции СДВИГ, например, как это показано на рис. 3.4, в список запретов добавляется элемент, состоящий из следующих операций: первая операция перемещает i-ю ячейку перестановки в позицию j, при условии, что в ней находится элемент B, а в j-й ячейке элемент D, и вторая, обратная, операция перемещает j-ю ячейку в позицию i, при условии, что она содержит элемент B, а i-я ячейка элемент C. При использовании операции ЗАМЕНА в список запретов также добавляются две операции, меняющие местами ячейки i и j, при условии, что в них содержатся элементы B и D. Длина списка запретов определяется параметром $l \geq 0$ и показывает максимальное количество элементов, которое может содержаться в списке. При добавлении нового элемента в список запретов в случае, когда его длина становится больше заданной, из него удаляется элемент, добавленный раньше всех. При $l = 0$ список запретов пуст и алгоритм превращается в стандартный алгоритм локального спуска, который совершает шаги только улучшающие целевую функцию и останавливается в локальном оптимуме. Выбор длины списка запретов зависит от размерности решаемой задачи и мощности окрестности. При коротком списке запретов алгоритм может зациклиться. При длинном списке, может оказаться так, что большая часть окрестности будет запрещена, что также не приведет к хорошим результатам. Ниже представлена общая схема алгоритма.

Вероятностный алгоритм поиска с запретами для задачи упаковки кругов и прямоугольников в полосу

1. Построить начальное решение i_0.
2. Положить $\mathrm{TABU}_l(i_0) := \emptyset$, $i^* := i_0$,$L^* := L(i^*)$, $i^r := i_0$, $L^r := L(i^r)$, $k := 0$, $P := P_{\min}$, $\mathrm{sgn} := 1$.
3. Повторять, пока не выполнен критерий остановки.
   - 3.1. Выполнить цикл N_{loop} раз.
     - 3.1.1. Сформировать окрестность $N_P(i_k)$.
     - 3.1.2. Найти i_{k+1} такое, что $L(i_{k+1}) = \min\limits_{j \in N_P(i_k) \setminus \mathrm{TABU}_l(i_k)} L(j)$.
     - 3.1.3. Если $L^* > L(i_{k+1})$, то $L^* := L(i_{k+1})$ и $i^* := i_{k+1}$.

3.1.4. Если $L^r > L(i_{k+1})$, то $L^r := L(i_{k+1})$, $i^r := i_{k+1}$ и $\mathrm{sgn} := 1$.

3.1.5. Положить $k := k + 1$ и обновить $\mathrm{TABU}_l(i_k)$.

3.2. Если $\mathrm{sgn} = 1$, то $i_k := i^r$.

3.3. Положить $P := P + \mathrm{sgn} \cdot \Delta P$.

3.4. Если $P \geq P_{\max}$, то $\mathrm{sgn} := -1$.

3.5. Если $P \leq P_{\min}$, то $\mathrm{sgn} := 1$, $L^r := L(i_k)$ и $i^r := i_k$.

4. Выдать результат i^* и L^*.

Параметры $P_{\min}$, $P_{\max}$ — верхняя и нижняя границы изменения параметра рандомизации, ΔP — величина изменения параметра рандомизации, l — длина списка запретов и N_{loop} — количество итераций цикла являются заданными. Выполнение алгоритма начинается с построения начального решения и присвоения начальных значений внутренним переменным, которыми являются k — номер итерации алгоритма, i_k — текущее решение, $\mathrm{TABU}_l(i_k)$ — список запретов на $k-$й итерации, P — параметр рандомизации окрестности, $\mathrm{sgn} \in \{-1, +1\}$ — указывает на убывание или возрастание параметра рандомизации, i^*, L^* — лучшее найденное решение и значение целевой функции для него и аналогично i^r, L^r — лучшее найденное решение при фиксированном значении параметра рандомизации окрестности и значение целевой функции для него. Далее происходит локальный поиск с запретами по рандомизированной окрестности $N_P(i) \backslash \mathrm{TABU}_l(i)$. Критерием качества решения выступает длина занятого участка полосы L. В процессе поиска параметр рандомизации P меняется в пределах $[P_{\min}, \ldots, P_{\max}]$ на величину ΔP в зависимости от того, как часто встречаются решения малой относительной погрешности. Таким способом осуществляется интенсификация и диверсификация поиска [35]. В начале работы алгоритма P принимает минимальное допустимое значение $P_{\min}$. Затем алгоритм выполняет заданное число N_{loop} шагов локального поиска при фиксированном значении рандомизации окрестности и запоминает наилучшее найденное решение для данного P. Это решение, в которое алгоритм возвращается при последующем увеличении значения параметра P. Таким образом, увеличивая значение P, а вместе с ним долю просматриваемой окрестности, и возвращаясь в наилучшее найденное на предыдущем этапе решение, реализуется интенсификация поиска. Другими словами, алгоритм осуществляет

более детальный поиск в окрестности лучших, ранее найденных решений. Действия повторяются до тех пор, пока значение параметра рандомизации не достигнет максимального значения $P_{\max}$. Затем доля просматриваемой окрестности начинает уменьшаться, и алгоритм переходит в фазу диверсификации, пытаясь для дальнейшего поиска перейти в новую область пространства решений. Если в процессе диверсификации удается найти решение лучше, чем i^r, то алгоритм начинает интенсификацию поиска, запомнив это решение в качестве i^r. В противном случае этап диверсификации длится до тех пор, пока параметр рандомизации P не достигнет минимального значения $P_{\min}$. Критерием остановки алгоритма служит либо время работы, либо число итераций, на протяжении которых алгоритму не удается улучшить значение целевой функции, либо общее число итераций.

3.6 Численные эксперименты

Разработанный алгоритм запрограммирован на языке PASCAL и тестировался на вычислительной машине с процессором Intel Celeron 1,8 GHz. Раздел вычислительных экспериментов состоит из трех частей. В первой части на примере задачи упаковки кругов в полосу изучается влияние параметра рандомизации окрестности и длины списка запретов на качество получаемых решений. Во второй части на основе полученных результатов для случайно сгенерированных примеров проводится анализ качества работы алгоритма путем сравнения с нижними и верхними оценками. Также здесь на известных тестовых примерах проводится сравнение с результатами, полученными ранее другими авторами для частных случаев рассматриваемой задачи. В третьей части описывается использование коммерческого пакета GAMS для решения задач упаковки и представлены полученные им результаты для некоторых из рассматриваемых примеров.

3.6.1 Влияние рандомизации и длины списка запретов

В [4] на примере многостадийной задачи размещения показано, что эффективность использования вероятностного локального поиска существенно зависит от значения параметра рандомизации окрестности и длины списка запретов. Алгоритму требуется настройка этих параметров для дальней-

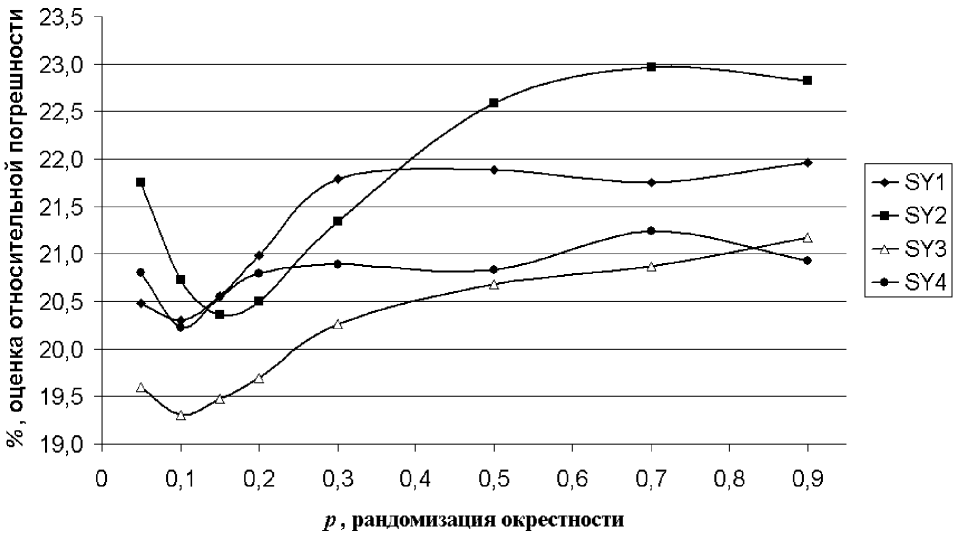


Рис. 3.5: Влияние рандомизации окрестности на относительную погрешность решений

ших вычислений. Для этого необходимо выяснить, как влияет размер доли рассматриваемой окрестности и длина списка запретов на качество получаемых решений. В результате численных расчетов удалось выяснить, что на примерах упаковки кругов алгоритм наиболее чувствителен к изменению этих параметров. Кроме того, данный класс примеров представляет особую сложность для алгоритма. Чем больше кругов содержится в примере, тем больше требуется времени, чтобы найти хороший локальный оптимум. Найти оптимальные параметры для всех классов входных данных не представляется возможным. Поэтому для настройки использовались примеры для задачи упаковки кругов в полосу, представляющих один из самых сложных классов.

В табл. 3.2 указаны характеристики используемых тестовых примеров: число кругов n_c, ширина полосы W и нижняя оценка оптимальной длины полосы LB, вычисленная по формуле $LB = \sum_{i=1}^{n} S_i/W$, где S_i — площадь i-го круга. Были взяты первые четыре примера из этой таблицы. Значение параметра рандомизации при каждом запуске алгоритма оставалось постоянным и в процессе поиска не менялось. Расчеты проводились для следующих значений $P = 0{,}05;\ 0{,}1;\ 0{,}15;\ 0{,}2;\ 0{,}3;\ 0{,}5;\ 0{,}7;\ 0{,}9$. Длина списка запретов l была постоянной и равнялась 20. Для каждого примера при каждом значении параметра P алгоритм применялся 10 раз. Время работы алгоритма при каждом запуске составляло 5 минут. На рис. 3.5 показана

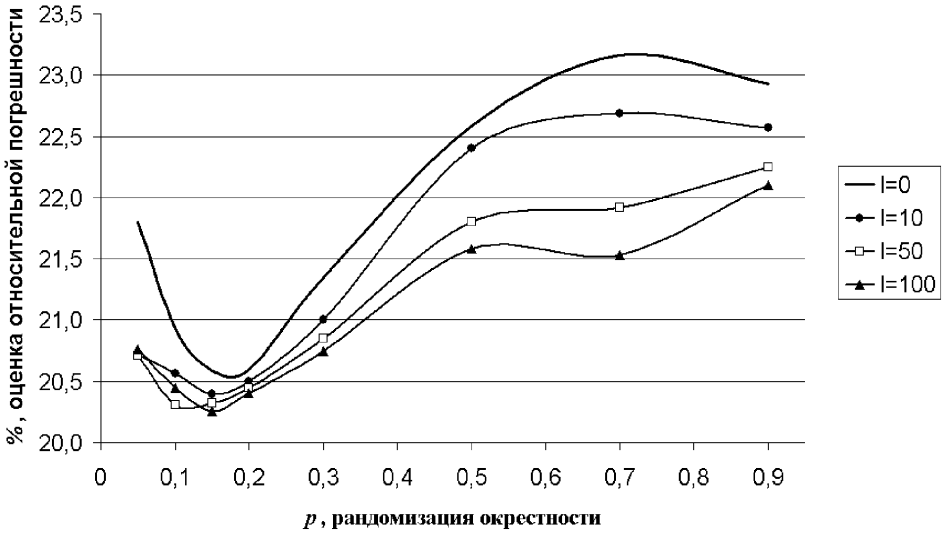


Рис. 3.6: Влияние списка запретов на относительную погрешность решений

зависимость оценки относительной погрешности от значения параметра P. Как видно из графиков, на данных примерах алгоритм достигает наибольшей эффективности при $P \in [0,05; 0,25]$. Стоит отметить, что чем больше рандомизация, тем меньшую часть окрестности алгоритм рассматривает на каждой итерации и тем меньше трудоемкость одной итерации. Следовательно, при большой рандомизации алгоритм вырождается в случайное блуждание, а при малой — поведение алгоритма будет похожим на стандартный локальный спуск.

Вторая часть эксперимента показывает влияние длины списка запретов на результаты работы алгоритма. В процессе эксперимента одновременно менялись значение параметра P=0,05; 0,1; 0,15; 0,2; 0,3; 0,5; 0,7; 0,9 и длина списка запретов l=0; 10; 50; 100. В качестве тестового примера использован пример SY2, содержащий 20 кругов. Для каждого набора параметров было произведено 30 запусков алгоритма, по 5 минут каждый. Результаты представлены на рис. 3.6. Несложно заметить, что длина списка запретов оказывает существенное влияние на процесс поиска, начиная со значений параметра рандомизации больших 0,3. При меньших значениях доли рассматриваемой окрестности влияние длины списка запретов уменьшается. Тем не менее, при его наличии алгоритм получает решения лучшего качества, чем без него.

3.6.2 Упаковка кругов и прямоугольников

Первый эксперимент заключается в анализе работы алгоритма на задаче упаковки кругов и прямоугольников в полосу. Несмотря на то, что данная задача рассматривалась ранее как частный случай в более общих постановках [37, 48, 66], в литературе для нее не приводится тестовых примеров. Поэтому примеры были сгенерированны случайным образом так, чтобы для каждого примера была известна некоторая верхняя оценка его оптимального решения. Генерация примеров осуществлялась следующим образом. Допустим, нам необходимо получить тестовый пример упаковки n предметов, n_c из которых круги, а n_r — прямоугольники, в полосу ширины W. Тогда мы разрезаем прямоугольную область размером $W \times UB$, где UB — заданная верхняя оценка оптимальной длины полосы, на n_c квадратов и n_r прямоугольников безотходным способом. Затем каждый квадрат заменяется на круг диаметром, равным стороне квадрата. Таким образом, если $n_c = 0$, то для такого примера существует решение, когда все прямоугольники упакованы в полосу длины UB безотходным способом, т. е. такое решение является оптимальным. В случае, когда $n_c > 0$, UB является лишь верхней оценкой оптимальной длины полосы. Всего было сгенерировано 24 примера с количеством предметов от 8 до 196. В табл. 3.1 указаны параметры всех примеров, включая верхнюю UB и нижнюю LB оценки.

В процессе эксперимента алгоритм вероятностного поиска с запретами отрабатывал на каждом примере десять раз. Был рассмотрен только неориентированный случай задачи, т. е. повороты прямоугольников на 90 градусов разрешались. Начальное решение выбиралось случайным образом. Параметр рандомизации менялся в диапазоне от $P_{\min}$ до $P_{\max}$, давая алгоритму возможность подстраиваться в процессе поиска. В предыдущем эксперименте было показано, что при рандомизации, равной 0,1, алгоритм способен за небольшое время найти достаточно хорошие решения, а начиная с 0,5, поведение алгоритма все больше становится похожим на стандартный локальный спуск. Поэтому значения $P_{\min}$ и $P_{\max}$ принимались равными 0,1 и 0,5, что позволило эффективно осуществлять интенсификацию и диверсификацию поиска.

Величина шага ΔP равнялась 0,05. Количество итераций, совершаемых алгоритмом при каждом значении P, бралось обратно пропорциональным

Таблица 3.1: Значение длины полосы для задачи упаковки кругов и прямоугольников

Пример	Параметры примера						TABU SEARCH		Отклонение (%)		Время
	W	n	n_c	n_r	LB	UB	Среднее	Лучшее	от LB	от UB	(сек.)
CR1P01	10	7	3	4	8,99	10	10,00	10,00	11,28	0,00	0
CR1P02	10	8	4	4	8,96	10	10,00	10,00	11,61	0,00	2
CR1P03	10	7	4	3	8,69	10	10,00	10,00	15,09	0,00	0
CR2P01	20	17	7	10	17,83	20	20,00	20,00	12,15	0,00	326
CR2P02	20	17	8	9	17,32	20	19,91	19,90	14,91	-0,50	417
CR2P03	20	17	6	11	17,69	20	20,00	20,00	13,04	0,00	346
CR3P01	40	25	10	15	13,46	15	15,27	15,27	13,42	1,77	738
CR3P02	40	25	6	19	13,83	15	16,17	15,81	14,34	5,42	596
CR3P03	40	25	14	11	13,18	15	15,00	15,00	13,80	0,00	920
CR4P01	60	29	10	19	27,61	30	30,97	30,49	10,43	1,64	597
CR4P02	60	29	14	15	26,21	30	30,43	30,00	14,45	0,00	671
CR4P03	60	29	6	23	28,45	30	31,00	31,00	8,94	3,33	558
CR5P01	60	49	20	29	53,14	60	63,04	61,61	15,93	2,68	904
CR5P02	60	49	16	33	53,96	60	63,21	62,40	15,64	4,00	810
CR5P03	60	49	12	37	56,99	60	63,58	63,00	10,54	5,00	996
CR6P01	60	73	39	34	78,30	90	93,15	91,93	17,42	2,15	1360
CR6P02	60	73	35	38	78,76	90	92,93	90,61	15,05	0,68	1027
CR6P03	60	73	31	42	78,98	90	93,98	92,34	16,92	2,60	1235
CR7P01	80	97	45	52	104,37	120	125,72	122,97	17,82	2,48	1220
CR7P02	80	97	41	56	106,87	120	126,03	125,75	17,68	4,80	1190
CR7P03	80	97	37	60	107,31	120	125,89	125,46	16,92	4,55	1065
CR8P01	160	196	80	116	212,23	240	255,71	251,92	18,70	4,97	1202
CR8P02	160	196	96	100	209,77	240	248,60	246,91	17,71	2,88	1612
CR8P03	160	196	110	86	202,65	240	239,44	235,77	16,34	-1,76	1508

размерности примера и равнялось $\lceil 1000/n \rceil$. Длина списка запретов была ограничена величиной $l = 50$. В качестве критерия остановки выступало время работы алгоритма, ограниченное тридцатью минутами, либо состояние, когда в результате прохождения параметра рандомизации значений от $P_{\min}$ до $P_{\max}$, не удавалось обновить лучшее найденное решение. В табл. 3.1 приведены результаты, полученные алгоритмом. В столбце TABU SEARCH для каждого примера указаны среднее и наименьшее значения длины полосы, найденные алгоритмом в результате десяти попыток. В следующих двух столбцах указано отклонение наименьшего значения длины полосы от нижней и верхней оценок. В последнем столбце приводится среднее время

работы алгоритма.

Численные эксперименты показывают, что для поставленной задачи разработанный алгоритм за приемлемое время получает решения близкие к нижним оценкам.

Цель следующих двух экспериментов заключалась в том, чтобы сравнить результаты работы разработанного алгоритма с результатами, полученными ранее другими авторами на известных тестовых примерах для двух задач, являющихся частными случаями поставленной задачи: упаковка кругов в полосу и упаковка прямоугольников в полосу [37, 40, 42, 66, 67].

Таблица 3.2: Значение длины полосы для задачи упаковки кругов

Пример	W	n_c	LB	CAGA [37]	S.Y. [66, 67]	B1.5 [42]	TABU SEARCH
SY1	9,5	30	14,55	18,37	17,49	17,29	17,26
SY2	8,5	20	12,16	15,24	14,89	14,53	14,51
SY3	9	25	12,23	15,41	14,93	14,47	14,43
SY4	11	35	19,91	24,98	24,35	23,55	23,50
SY5	15	100	31,28	38,85	38,05	36,33	36,71
SY6	19	100	31,78	39,65	38,65	36,86	37,42

Для задачи упаковки кругов рассмотрены шесть тестовых примеров из электронной библиотеки

http://www.laria.u-picardie.fr/hifi/OR-Benchmark/TDL/.

Их размерность меняется от 20 до 100. Работа алгоритма вероятностного поиска с запретами на данных примерах выполнялась с теми же параметрами, что и в предыдущем эксперименте. Однако алгоритм при каждом запуске полностью использовал отведенное ему время, и из двух критериев остановки всегда срабатывало только ограничение по времени. В табл. 3.2 представлены решения генетического алгоритма CAGA [37], решения, полученные методом ветвей и границ S.Y. [66, 67], а также решения, полученные жадной эвристической процедурой B1.5 [42]. Время работы генетического алгоритма составляло 30 минут на компьютере с процессором Pentium III 733 MHz. Время работы метода ветвей и границ не указано, за исключением примера SY6, для которого потребовался один час на IBM PC/AT 486, чтобы получить решение, указанное в табл. 3.2. Среднее время

работы алгоритма B1.5 составляет примерно 40 минут для примеров SY1–SY4 и 18 часов для примеров SY5 и SY6 на машине с процессором Athlon XP2000+. При этом стоит отметить, что последний алгоритм разработан для решения задачи упаковки кругов в прямоугольный контейнер, и за указанное время ему удавалось разместить все круги в контейнере, указанного размера. Время работы вероятностного поиска с запретами, как уже было сказано, составляло 30 минут.

Из таблицы видно, что для первых четырех примеров SY1–SY4 алгоритму удалось найти новые рекордные решения. Для двух оставшихся наилучшими найденными остаются решения, полученные алгоритмом B1.5.

Таблица 3.3: Тестовые примеры для задачи упаковки прямоугольников в полосу

Класс (Примеры)	Количество предметов n_r	Оптимальное решение $W \times L^*$
C1(C11,C12,C13)	16(C11,C13),17(C12)	20×20
C2(C21,C22,C23)	25(C21,C22,C23)	40×15
C3(C31,C32,C33)	28(C31,C33),29(C32)	60×30
C4(C41,C42,C43)	49(C41,C42,C43)	60×60
C5(C51,C52,C53)	73(C51,C52,C53)	60×90
C6(C61,C62,C63)	97(C61,C62,C63)	80×120
C7(C71,C72,C73)	196(C71,C73),197(C72)	160×240

Для задачи упаковки прямоугольников в полосу были рассмотрены примеры с известным оптимальным значением целевой функции из работы [40]. Все примеры поделены на семь классов в соответствии с их размерностью — по три примера в каждом классе. Оптимальная длина полосы для каждого примера известна и равна L^* (табл. 3.3). Рассмотрены два варианта задачи: с поворотами прямоугольников на 90 градусов и без них. В табл. 3.4 и 3.5 для данных примеров указаны лучшие результаты, полученные описанными в литературе алгоритмами [12, 13, 19, 20, 40, 41, 44, 45, 75], а также результаты, полученные разработанным алгоритмом. Все они представляют собой процентное отклонение от известного оптимального решения. Алгоритм поиска с запретами, как и в предыдущих экспериментах, запускался на каждом примере 10 раз, и затем выбиралось лучшее найденное решение. Начальные значения параметров и критерии остановки не менялись. В табл. 3.4 и 3.5 также указано среднее время работы

алгоритма на примерах из каждого класса.

Рассмотрим задачу с поворотами прямоугольников. В табл. 3.4 представлены результаты, опубликованные в литературе [13, 19, 40, 41, 75] и полученные разработанным алгоритмом. Генетический алгоритм GA+BLF и алгоритм имитации отжига SA+BLF [40] работали на вычислительной машине с процессором Pentium Pro 200 Mhz, и среднее время запуска на каждом примере составляло 674 минуты для SA+BLF и 136 минут для GA+BLF. Время работы алгоритма SPGAL [19] составляло 160 секунд на вычислительной машине с процессором Pentium 2 GHz и алгоритм запускался 10 раз на каждом примере. Алгоритм GA+IHR [75] тестировался на вычислительной машине с тактовой частой процессора 2,4 GHz, и его среднее время работы составляло 76,55 секунд (0,88 секунд для примеров из класса C1 и 426,04 для примеров из класса C7). Среднее время работы жадного эвристического алгоритма HRP на машине с процессором Athlon XP2000+ составляло около 13 минут (от нескольких секунд для небольших примеров до одного часа для примеров с большим числом предметов). При этом алгоритм HRP представляет собой аналогичную версию алгоритма B1.5 из предыдущего эксперимента и также был разработан для решения задачи упаковки в прямоугольный контейнер. Указанное время алгоритм использовал для размещения всех предметов в контейнере заданного размера. Алгоритм H-SP_{1000s} [13] для каждого примера запускался 10 раз по 1000 секунд.

Таблица 3.4: Относительная погрешность для неориентированной упаковки прямоугольников в полосу

Класс	GA+BLF [40]	SA+BLF [40]	SPGAL [19]	GA+IHR [75]	HRP [41]	H-SP_{1000s} [13]	TABU SEARCH	Время (сек.)
C1	4	4	1,7	3,33	0	0	0	15
C2	7	6	0	4,44	0	0	0	34
C3	5	5	2,2	2,22	0	1,11	2,22	12
C4	3	3	0	1,67	0	1,67	1,67	47
C5	4	3	0	1,11	0	1,11	1,85	364
C6	4	3	0,3	0,83	0,83	0,83	2,5	135
C7	5	4	0,3	0,83	0,83	0,56	2,64	943
Среднее	4,57	4	0,64	2,06	0,24	0,75	1,55	221

В табл. 3.5 показаны результаты, полученные для ориентированной за-

дачи прямоугольной упаковки в полосу [12, 13, 19, 20, 44, 45]. Алгоритм Iori [45] работал 300 секунд на каждом примере на машине с процессором Pentium III 800 Mhz. BF+SA [20] и GRASP [12] запускались 10 раз для каждого примера на компьютере Pentium 2 Ghz с ограничением по времени — 60 секунд на один запуск. Условия работы алгоритмов SPGAL [19] и H-SP$_{1000s}$ [13] для ориентированной задачи были такими же, как для неориентированной. Среднее время работы метаэвристики HM-SP [44] составляло 756 секунд на процессоре Pentium III 1 GHz (10 секунд для примеров из класса C1 и 3600 секунд для примеров из класса C7).

Как видно из табл. 3.4 и 3.5, алгоритму, разработанному для более общей задачи, удалось получить неплохие решения для примеров задачи прямоугольной упаковки. Полученные результаты показывают, что алгоритм вероятностного поиска с запретами незначительно уступает, а в некоторых случаях даже превосходит специализированные на прямоугольной упаковке алгоритмы. Кроме того, для всех примеров из первых двух классов разработанному алгоритму всегда удавалось находить оптимальные решения задачи.

Таблица 3.5: Относительная погрешность для ориентированной упаковки прямоугольников в полосу

Класс	Iori [45]	BF+SA [20]	SPGAL [19]	HM-SP [44]	GRASP [12]	H-SP$_{1000s}$ [13]	TABU SEARCH	Время (сек.)
C1	1,67	0	1,67	2,44	0	0	0	12
C2	2,22	6,25	2,22	6,25	0	0	0	21
C3	2,22	3,33	3,33	3,33	1,11	2,22	2,22	18
C4	4,75	1,67	2,78	3,12	1,67	1,67	2,22	47
C5	3,93	1,48	1,48	4,98	1,11	1,11	2,22	124
C6	4,00	1,39	1,67	3,15	0,83	0,83	2,5	117
C7	—	1,77	1,25	3,45	1,25	0,97	2,78	815
Среднее	3,13	2,27	2,06	3,82	0,88	0,97	1,7	165

3.6.3 Использование пакета GAMS

В данном разделе приводятся результаты, полученные с помощью коммерческого пакета GAMS. Использовались тестовые примеры, рассмотренные ранее. Для работы на каждом примере пакету отводилось 20 часов. В ка-

честве решающего ядра для задач с квадратичными ограничениями был выбран решатель BARON (версия 8.1.1). Задачи с линейными ограничениями решались с помощью CPLEX (версия 11.0).

Неориентированная задача упаковки кругов и прямоугольников в полосу, математическая постановка которой представлена в начале работы, оказалась для GAMS слишком сложной. В результате, за отведенное время пакету удалось найти оптимальные решения для CR1P1, CR1P2 и CR1P3. Для остальных примеров не получено ни одного допустимого решения. Что касается частных случаев задачи, то были получены результаты для четырех примеров упаковки кругов SY1–SY4 и для примеров ориентированной упаковки прямоугольников из первых пяти классов C1–C5. При решении последних двух задач пакетом использовались математические постановки, выписанные ниже. Они имеют более простой вид, чем общая. Первая постановка для задачи упаковки кругов:

найти $\min L$ при ограничениях:

$$
\begin{aligned}
& x_i^c - r_i \geq 0, && i \in I,\\
& y_i^c - r_i \geq 0, && i \in I,\\
& x_i^c + r_i \leq L, && i \in I,\\
& y_i^c + r_i \leq W, && i \in I,\\
& \left(x_i^c - x_k^c\right)^2 + \left(y_i^c - y_k^c\right)^2 \geq (r_i + r_k)^2, && i, k \in I.
\end{aligned}
$$

Задача, представленная в таком виде относится к классу задач квадратичного программирования и уже является достаточно сложной для нахождения глобального оптимума.

Следующая постановка описывает задачу ориентированной упаковки прямоугольников в полосу:

найти $\min L$ при ограничениях:

$$
\begin{aligned}
& x_j^r \geq 0, && j \in J,\\
& y_j^r \geq 0, && j \in J,\\
& x_j^r + l_j \leq L, && j \in J,\\
& y_j^r + w_j \leq W, && j \in J,\\
& x_j^r + l_j \leq x_m^r + (1 - a_{jm}^L) \sum_{s=1}^{n_r} l_s, && j, m \in J,
\end{aligned}
$$

$$y_j^r + w_j \le y_m^r + (1 - a_{jm}^B)W, \qquad j, m \in J,$$
$$a_{jm}^L + a_{mj}^L + a_{jm}^B + a_{mj}^B = 1, \qquad j, m \in J.$$

В связи с тем, что в задаче используются бинарные переменные a_{jm}^L и a_{jm}^B, а все ограничения линейны, она переходит в класс задач частично-целочисленного линейного программирования.

Таблица 3.6: Результаты, полученные пакетом GAMS

Задача	Пример	GAMS	Оптимум	TABU SEARCH	Оптимум
Неориентированная упаковка кругов и прямоугольников	CR1P1	10	да	10	да
	CR1P2	10	да	10	да
	CR1P3	10	да	10	да
Упаковка кругов	SY1	19,85	нет	17,26	неизвестно
	SY2	17,72	нет	14,51	неизвестно
	SY3	16,76	нет	14,43	неизвестно
	SY4	25,89	нет	23,50	неизвестно
Ориентированная упаковка прямоугольников	C1P1	20	да	20	да
	C1P2	20	да	20	да
	C1P3	20	да	20	да
	C2P1	16	нет	15	да
	C2P2	16	нет	15	да
	C2P3	16	нет	15	да
	C3P1	32	нет	31	нет
	C3P2	32	нет	31	нет
	C3P3	31	нет	30	да
	C4P1	71	нет	62	нет
	C4P2	68	нет	61	нет
	C4P3	67	нет	61	нет
	C5P1	121	нет	92	нет
	C5P2	120	нет	92	нет
	C5P3	134	нет	92	нет

В табл. 3.6 представлены результаты, полученные с помощью пакета GAMS. Оптимальные решения, найденные для примеров CR1P1–CR1P3 задачи упаковки кругов и прямоугольников, совпадают с решениями, полученными алгоритмом поиска с запретами. Для примеров C1P1–C1P3 задачи ориентированной упаковки прямоугольников пакетом также были найдены оптимальные решения, однако доказать их оптимальность за отведенное время не удалось. Качество допустимых решений, полученных для

остальных примеров, заметно уступает решениям, полученным с помощью вероятностного поиска с запретами за гораздо меньшее время.

3.7 Выводы к главе 3

Рассмотрена задача упаковки кругов и прямоугольников в полосу. Выписаны математические постановки для общего и частных случаев задачи. Для решения задачи разработан алгоритм вероятностного поиска с запретами, использующий двухконтактную схему кодирования решений. Эта схема использует оригинальную процедуру декодирования, восстанавливающую упаковку предметов по заданному коду. Алгоритм осуществляет локальный поиск в пространстве двухконтактных решений с использованием процедур интенсификации и диверсификации, основанных на адаптивном изменении параметра рандомизации окрестности. Численные эксперименты показали, что алгоритм позволяет находить решения с малой погрешностью, в том числе и глобально оптимальные на примерах небольшой размерности. На известных тестовых примерах для частных случаев задачи алгоритм нашел новые рекордные значения целевой функции для четырех примеров упаковки кругов в полосу. Представляет интерес разработка новой, менее трудоемкой схемы кодирования, которая за фиксированное время позволит выполнить большее количество итераций. В качестве одного из вариантов интересно рассмотреть декодер с использованием контура частичной упаковки предметов. Подобная идея рассматривалась в работе [15] для задачи прямоугольной упаковки в контейнеры с запрещенными областями. Это бы позволило сократить трудоемкость декодирования на порядок и существенно ускорить работу алгоритма локального поиска, особенно, на примерах большой размерности.

Заключение

1. Для решения задачи упаковки прямоугольников в прямоугольную область минимальной площади предложен гибридный алгоритм имитации отжига, использующий новую процедуру уплотнения упаковки. В ходе вычислительных экспериментов с помощью разработанного алгоритма были найдены новые рекордные значения целевой функции для семи примеров из электронных библиотек MCNC и GSRC с числом предметов от 33 до 300.

2. Исследован новый класс задач прямоугольной упаковки в контейнеры с запрещенными областями. Получены математические модели в терминах частично-целочисленного программирования. Для решения задач разработаны новые кодирующие схемы. На их основе разработан модифицированный алгоритм имитации отжига, позволяющий решать четыре типа задач с запрещенными областями. Экспериментально установлено, что алгоритм позволяет находить решения с малой погрешностью, в том числе и оптимальные решения на примерах с числом предметов до 20.

3. Для задачи упаковки кругов и прямоугольников в полосу разработана оригинальная процедура декодирования, восстанавливающая по заданной перестановке двухконтактную упаковку предметов. Разработан алгоритм вероятностного поиска с запретами с адаптивно изменяемой рандомизацией окрестности. В результате численных экспериментов для четырех известных примеров упаковки кругов в полосу получены новые рекордные значения целевой функции.

Список литературы

[1] Абгарян К. К., Хачатуров В. Р. Компьютерное моделирование устойчивых структур кристаллических материалов // Журнал вычислительной математики и математической физики — 2009. — Т. 49, № 8. — С. 1517—1530.

[2] Вайнштейн А. Д. Задачи об упаковке прямоугольников в полосу (обзор) // Управляемые системы. — 1984. — Вып. 25. — С. 17—37.

[3] Валеева А. Ф., Сиразетдинова Т. Ю. Алгоритмы решения задачи прямоугольного гильотинного раскроя на базе метаэвристики имитации отжига // Материалы конференции «Проблемы оптимизации и экономические приложения». — Омск: Изд-во ОмГТУ, 2006. — С. 126.

[4] Гончаров Е. Н., Кочетов Ю. А. Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискретный анализ и исследование операций. Сер. 2. — 2002. — Т. 9, № 2. — С. 13—30.

[5] Гэри М. Р., Джонсон Д. С. Вычислительные машины и труднорешаемые задачи. — Москва: Мир, 1982. — 416 с.

[6] Кочетов Ю. А. Вероятностные методы локального поиска для задач дискретной оптимизации // Дискретная математика и ее приложения. Сборник лекций молодежных и научных школ по дискретной математике и ее приложениям. — М: МГУ, 2001, — С. 87–117.

[7] Кочетов Ю., Младенович Н., Хансен П. Локальный поиск с чередующимися окрестностями // Дискретый анализ и исследование операций. Сер. 2. — 2003. — Т. 10, № 1. — С. 11—43.

[8] Мухачева Э. А. Обзор и перспективы развития комбинаторных методов решения задач раскроя и упаковки // Материалы конференции

«Дискретный анализ и исследование операций». — Новосибирск: Изд-во Ин-та математики, 2002. — С. 80–87.

[9] Стоян Ю. Г., Яковлев С. В. Математические модели и оптимизационные методы геометрического проектирования. — Киев: Наукова думка, 1986. — 266 с.

[10] Стрекаловский А. С. Элементы невыпуклой оптимизации. — Новосибирск: Наука, 2003. — 352 с.

[11] Хачатуров В. Р., Абгарян К. К., Бакаев А. В., Галиулин Р. В., Кабанович В. И. Перебор всевозможных кристаллических структур для заданной химической формулы и симметрии на примере перовскита // Математическое моделирование композиционных объектов. — Москва: Изд-во ВЦ РАН, 1997. — Вып. 2. — С. 6—22.

[12] Alvarez-Valdes R., Parreno F., Tamarit J. M. Reactive GRASP for the strip packing problem // Comput. Oper. Res. — 2008. — Vol. 35, N 4. — P. 1065–1083.

[13] Araya I., Neveu B., Riff M. C. An efficient hyperheuristic for strip-packing problems // Studies in Comput. Intelligence. — 2008. — Vol. 136. — P. 61–76.

[14] Baker B. S., Coffman E. G., Rivest R. L. Orthogonal packing in two dimensions // SIAM J. Compututing. — 1980. Vol. 9, N 4. — P. 846–855.

[15] Beisiegel B., Kallrath J., Kochetov Y., Rudnev A. Simulated annealing based algorithm for the 2D bin packing problem with impurities // Proc. Operations Research, 2005. — Heidelberg: Springer, 2006. — P. 109–113.

[16] Birgin E. G., Martinez J. M., Nishihara F. H, Roncony D. P. Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization // Comput. Oper. Res. — 2006. — Vol. 33. — P. 3535–3548.

[17] Boschetti M. A., Mingozzi A. The two-dimensional finite bin packing problem. Part I: New lower bounds for the oriented case // 4OR. — 2003. – Vol. 1. — P. 27–72.

[18] Boschetti M. A., Mingozzi A. The two-dimensional finite bin packing problem. Part II: New lower and upper bounds // 4OR. — 2003. — Vol. 1. — P. 135–147.

[19] Bortfeldt A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces // Europ. J. Oper. Res. — 2006. Vol. 172, N 3. — P. 814–837.

[20] Burke E., Kendall G., Whitwell G. Metaheuristic enhancements of the best-fit heuristic for the orthogonal stock cutting problem // Computer Science Technical Report No. NOTTCS-TR-2006-3, University of Nottingham, 2006. — 30 p.

[21] Chan H. H., Markov I. L. Practical slicing and non-slicing block-packing without simulated annealing // Proc. Great Lakes Symposium on VLSI, 2004. — New York: ACM, 2004. — P. 282–287.

[22] Chang Y. C., Chang Y. W., Wu G. M., Wu S. W. B*-trees: A new representation for non-slicing floorplan // Proc. DAC, 2000. — New York: ACM, 2000. — P. 458–463.

[23] Chazelle B. The bottom-left bin packing heuristic: An efficient implementation // IEEE Trans. Compututing. — 1983. Vol. 32. — P. 697–707.

[24] Chung F. R. K., Garey M. R., Johnson D. S. On packing two-dimensional bins // SIAM J. Algebraic Discrete Meth. — 1986. Vol. 3. — P. 66–76.

[25] Coffman E. G., Garey M. R., Johnson D. S, Tarjan R. E. Performance bounds for level-oriented two-dimensional packing algorithms // SIAM J. Compututing. — 1980. Vol. 9. — P. 801–826.

[26] Dell'Amico M., Martello S., Vigo D. A lower bound for the non-oriented two-dimensional bin packing problem // Discrete Appl. Math. — 2002. — Vol. 118. — P. 13–24.

[27] Dong S., Hong X., Qi X., Wang R., Chen S., Gu J. VLSI module placement with pre-placed modules and considering congestion using

solution space smoothing // Proc. ASP-DAC, 2003. — New York: ACM, 2003. — P. 741–744.

[28] Dong S., Hong X., Wu Y., Xiu Z., Gu J. VLSI placement with pre-placed modules based on less flexibility first principles // Proc. ASIC, 2001. — Beijing: IEEE Press , 2001. — P. 106–109.

[29] Dongarra J. J. Performance of various computers using standard linear equations software // Technical Report No. CS-89-85, University of Manchester, 2008. — 102 p.

[30] Dowsland K. A., Dowsland W. B. Packing problems // Europ. J. Oper. Res. — 1992. — Vol. 56. — P. 2–14.

[31] Dreo J., Petrowski A., Siarry P., Taillard E. Metaheuristics for hard optimization: methods and case studies. — Berlin: Springer–Verl., 2005. — 369 p.

[32] Fekete S. P., Schepers J. On more-dimensional packing II: Bounds // Technical Report No. 97.289, Universität zu Köln, 2000. — 20 p.

[33] Gardner M. Some packing problems that cannot be solved by sitting on the suitcase // Scientific American — 1979. — Vol. 241, N 4. — P. 22–26.

[34] George J. A., George J. M., Lamar B. W. Packing different-sized circles into a rectangular container // Europ. J. Oper. Res. — 1995. — Vol. 84. — P. 693–712.

[35] Glover F., Laguna M. Tabu search. — Dordrecht: Kluwer Acad. Publ., 1997. — 382 p.

[36] Guo P. N., Cheng C. K., Yoshimura T. An O-tree representation of non-slicing floorplan and its applications // Proc. DAC, 1999. — New York: ACM, 1999. — P. 268–273.

[37] Hifi M., M'Hallah R. A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes // Internat. Transaction in Oper. Res. — 2003. — Vol. 10. — P. 195–216.

[38] Hifi M., M'Hallah R. Approximate algorithms for constrained circular cutting problems. // Comput. Oper. Res. — 2004. — Vol. 31. — P. 675–694.

[39] Hong X., Huang G., Cai Y., Gu J., Dong S., Cheng C. K., Gu J. Corner Block List: An effective and efficient topological representation of non-slicing floorplan // Proc. ICCAD, 2000. — Piscataway: IEEE Press, 2000. — P. 8–12

[40] Hopper E., Turton B. C. H. An empirical investigation of metaheuristic and heuristic algorithms for a 2D packing problem // Europ. J. Oper. Res. — 2001. — Vol. 128. — P. 34–57.

[41] Huang W., Chen D., Xu R. A new heuristic algorithm for rectangle packing // Comput. Oper. Res. — 2007. — Vol. 34. — P. 3270–3280.

[42] Huang W. Q., Li Y., Akeb H., Li C. M. Greedy algorithms for packing unequal circles into a rectangular container // J. Oper. Res. Soc. — 2005. — Vol. 56. — P. 539–548.

[43] Huang W. Q., Li Y., Li C. M., Xu R. C. New heuristics for packing unequal circles into a circular container // Comput. Oper. Res. — 2006. — Vol. 33. — P. 2125–2142.

[44] Ibaraki T., Imahori S., Yagiura M. Hybrid metaheuristics for packing problems // Studies in Comput. Intelligence. — 2008. — Vol. 114. — P. 185–219.

[45] Iori M., Martello S., Monaci M. Metaheuristic algorithms for the strip packing problem, — Dordrecht: Kluwer Acad. Publ., 2003. — P. 159—179.

[46] Johnson D. S., Aragon C. R., McGeoch L. A., Schevon C. Optimization by simulated annealing: An experimental evaluation, part I (graph partitioning) // INFORMS Oper. Res. — 1989. — Vol. 7, N 6. — P. 865–891.

[47] Johnson D. S., Demers A., Ullman J. D., Garey M. R., Graham R. L. Worst-case performance bounds for simple one-dimensional packing algorithms // SIAM J. Compututing. — 1974. Vol. 3. — P. 299–325.

[48] Kallrath J. Cutting circles and polygons from area-minimizing rectangles // J. Global Optimization. — 2009. — Vol. 43. — P. 299–328.

[49] Kallrath J., Kochetov Y., Rudnev A. Strip packing problem for circles and rectangles // 4th ESICUP Meeting. — Tokyo: University of Tokyo, 2007. — P. 20.

[50] Kenmochi M., Imamichi T., Nonobe K., Yagiura M., Nagamochi H. Exact algorithms for two-dimensional strip packing problem with and without rotations // Europ. J. Oper. Res. — 2009. — Vol. 198. — P. 73–83.

[51] Kirkpatrick S., Gelatt C., Vecchi M. Optimization by simulated annealing // Science. — 1983. — Vol. 220. — P. 671–680.

[52] Lin J. M., Chang Y. W. Corner sequence — a P–admissible floorplan representation with a worst case linear-time packing scheme // IEEE Trans. Very Large Integration Systems. — 2003. — Vol. 11, N 4. — P. 679–686.

[53] Lin J. M., Chang Y. W. TCG: A transitive closure graph-based representation for non-slicing floorplans // Proc. DAC, 2001. — New York: ACM, 2001. — P. 764–769.

[54] Lodi A., Martello S., Monaci M. Two-dimensional packing problems: A survey // Europ. J. Oper. Res. — 2002. — Vol. 141. — P. 241–252.

[55] Lodi A., Martello S., Vigo D. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems // INFORMS J. Computing. — 1999. — Vol. 11. — P. 345–357.

[56] Lodi A., Martello S., Vigo D. Recent advances on two-dimensional bin packing problems // Discrete Appl. Math. — 2002. — Vol. 123/124. — P. 373–380.

[57] Lubachevsky B. D., Graham R. Dense packing of congruent circles in rectangles with a variable aspect ratio. Discrete and computational geometry — the Goodman-Pollack festschrif // Algorithms and combinatorics. — Heidelberg: Springer–Verl., 2003. — P. 633–650.

[58] Martin O. C., Otto S. W. Combining simulated annealing with local search heuristics // Technical Report No. CSE-94-016, Oregon Graduate Institute School of Science and Engineering, 1993. — 15 p.

[59] Murata H., Fujiyoshi K., Nakatake S., Kajitani Y. VLSI module placement based on rectangle-packing by the sequence-pair // IEEE Trans. Computer Aided Design. — 1996. — Vol. 15, N 12. — P. 1518–1524.

[60] Murata H., Kuh E. S. Sequence-pair based placement method for hard/soft/pre-placed modules // Proc. international symposium on Physical design, 1998. — New York: ACM, 1998. — P. 167–172.

[61] Nakatake S., Fujiyoshi K., Murata H., Kajitani Y. Module placement on BSG - structure and IC layout applications // Proc. ICCAD, 1996. — Washington: IEEE Computer Society, 1996. — P. 484–491

[62] Ortmann F. G., Ntene N., van Vuuren J. H. New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems // Europ. J. Oper. Res. — 2010. — Vol. 203, N 2. — P. 306–315.

[63] Osman I. H., Laporte G. Metaheuristics: A bibliography // Ann. Oper. Res. — 1996. — Vol. 63 — P. 513–628.

[64] Pisinger D., Sigurd M. The two-dimensional bin packing problem with variable bin sizes and costs // Discrete Optimization. — 2005. — Vol. 2, N 2. — P. 154–167.

[65] Sakanushi K., Kajitani Y., Mehta D. P. The quarter-state-sequence floorplan representation // IEEE Trans. Circuits and Systems. — 2003. — Vol. 50. — P. 376–386.

[66] Stoyan Y. G., Yaskov G. N. Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints // Int. Trans. Oper. Res. — 1998, — Vol. 5, N 1. — P. 45–57.

[67] Stoyan Y. G., Yaskov G. N. A mathematical model and a solution method for the problem of placing various-sized circles into a strip // Europ. J. Oper. Res. — 2004. — Vol. 156. — P. 590–600.

[68] Takahashi T. A new encoding scheme for rectangle packing problem // Proc. ASP–DAC, 2000. — New York: ACM, 2000. — P. 175–178.

[69] Tang X., Tian R., Wong D. F. Fast evaluation of sequence pair in block placement by longest common subsequence computation // IEEE Trans. Computer Aided Design of Integrated Circuits and Systems. — 2001. — Vol. 20. — P. 1406–1413.

[70] Tang X., Tian R., Wong D. F. Fast-SP: A fast algorithm for block placement based on sequence-pair // Proc. ASP–DAC, 2001. — New York: ACM, 2001. — P. 521–526.

[71] Wong D. F., Liu C. L. A new algorithm for floorplan design // Proc. DAC, 1986. — Piscataway: IEEE Press, 1986. — P. 101–107.

[72] Wu Y. L., Huang W., Lau S., Wong C. K., Young G. H. An effective quasi-human based heuristic for solving the rectangle packing problem // Europ. J. Oper. Res. — 2002. — Vol. 141. — P. 341–358.

[73] Yong F. Y., Wong D. F. Slicing floorplans with pre-placed modules // Proc. ICCAD, 1998. — New York: ACM, 1998. — P. 252–258.

[74] Yu H. X., Zhang L. W. A nonlinear programming model for the packing of unequal circles into a square box // Proceedings of the 6th World Congress on intelligent control and automation. — Dalian: IEEE Robotics and Automation Society, 2006. — P. 1044–1047.

[75] Zhang D. F., Chen S. D., Liu Y. J. An improved heuristic recursive strategy based on genetic algorithm for the strip rectangular packing problem // Automatica Sinica. — 2007. — Vol. 33, N 9. — P. 911–916.